Improved Implicit Neural Representation with Fourier Reparameterized Training

Kexuan Shi Xingyu Zhou Shuhang Gu*
School of Computer Science and Engineering, UESTC
{kexuanshi712, xy.chous526, shuhanggu}@gmail.com

Abstract

Implicit Neural Representation (INR) as a mighty representation paradigm has achieved success in various computer vision tasks recently. Due to the low-frequency bias issue of vanilla multi-layer perceptron (MLP), existing methods have investigated advanced techniques, such as positional encoding and periodic activation function, to improve the accuracy of INR. In this paper, we connect the network training bias with the reparameterization technique and theoretically prove that weight reparameterization could provide us a chance to alleviate the spectral bias of MLP. Based on our theoretical analysis, we propose a Fourier reparameterization method which learns coefficient matrix of fixed Fourier bases to compose the weights of MLP. We evaluate the proposed Fourier reparameterization method on different INR tasks with various MLP architectures, including vanilla MLP, MLP with positional encoding and MLP with advanced activation function, etc. The superiority approximation results on different MLP architectures clearly validate the advantage of our proposed method. Armed with our Fourier reparameterization method, better INR with more textures and less artifacts can be learned from the training data. The codes are available at https://github.com/LabShuHangGU/FR-INR.

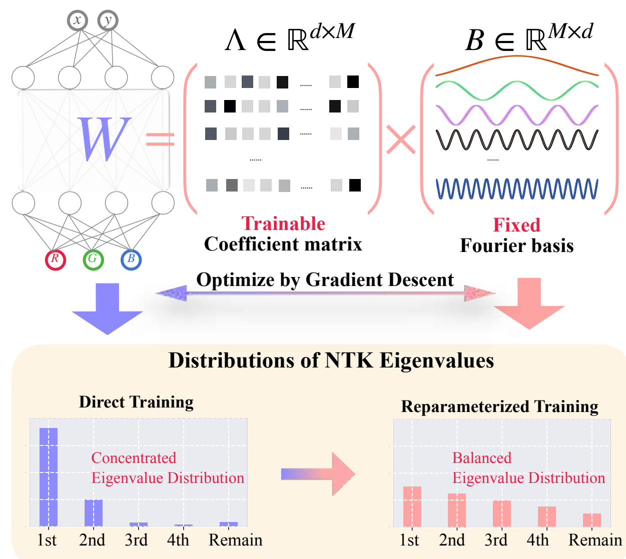

Figure 1. A conceptual illustration of our Fourier reparameterization method. We reparameterize the linear weight $\mathbf{W}$ with a trainable coefficient matrix $\boldsymbol{\Lambda}$ and a fixed Fourier basis matrix $\mathbf{B}$. More balanced eigenvalue distribution of neural tangent kernel (NTK) matrix implies that our method is able to alleviate the low-frequency bias of deep neural network, and therefore leads to better implicit neural representation.

1. Introduction

Recently, a novel signal representation paradigm called implicit neural representation (INR) has gained great attention in the field of computer vision and graphics. The main idea of INR is using multi-layer perceptron (MLP) to parameterize continuous and differentiable functions in an implicit manner. For example, given a gray scale image, INR takes the coordinates of the pixels as inputs to MLP and trains it to output the exact gray values using gradient-based optimization methods. Benefiting from the continuity nature and the expressive power of MLP, a more versatile continuous representation can be learned than traditional discrete grid-based methods. Consequently, INR has achieved state-of-the-art performance across a variety of tasks such as signal representation [34, 35], 3D shape reconstruction [3, 6, 30] and novel view synthesis [26, 28, 37].

Despite the universal approximation capabilities of MLP which have been proved in [16], obtaining highly accurate INR is not trivial. Specifically, MLP with ReLU activation function often fails to represent the high-frequency components of the signal such as the complex texture information in images and the intricate geometric shapes involved in 3D shape reconstruction. Such tendency of MLP to learn simple patterns of the target function is referred to as spectral bias [31] or low-frequency bias [43]. To improve the performance of INR, great efforts have been made to alleviate or circumvent the spectral bias of MLP. One main category

*Corresponding author

of approaches find that the difficulty of learning high frequencies becomes easier by increasing the complexity of the input data manifold [31] and explicitly extracting high-frequency features (for example, positional encoding [26]) to deal with the spectral bias issue. While, inspired by the pioneer work of [35], another main class of approaches exploit advanced activation functions [13, 34, 45] for pursuing more accurate approximation.

In this paper, we dive into details of the spectral bias issue and prove that reparameterizing weights of MLP with appropriate bases could provide us with a chance to narrow the gap between the magnitude of low-frequency and high-frequency loss, *i.e.* to alleviate the spectral bias during the training of deep neural networks. We propose a Fourier reparameterization strategy (see Fig. 1) and evaluate our method on simple 1D function approximation task and several real-world vision applications. Experimental results clearly validate our advantages in alleviating the low-frequency bias issue for improving approximation accuracy. Our study provides the literature with a practical reparameterization solution for improving the approximation accuracy of MLP without modifying its network architecture, previously, which so far has only been studied for the convolutional architectures. Moreover, our theoretical analysis sheds new light on the advantage of reparameterized training by connecting it with the spectral bias issue. We hope our study could inspire future works in improving training dynamics of deep neural networks with sophisticated reparameterization methods.

Our contributions are summarized as follows:

- We connect network training bias with reparameterization technique and theoretically prove that appropriate reparameterization could alleviate the spectral bias issue by altering the magnitude of gradients from different frequencies.
- We propose a practical reparameterization method for multi-layer perceptron, *i.e.* the Fourier reparameterization scheme, which could effectively improve the approximation accuracy of implicit neural representation without modifying its network architecture.
- We provide detailed experimental analysis on a wide range of implicit neural representation tasks. Our Fourier reparameterization method allows the improvement of commonly used network architectures and provides an representation with more high-frequency details.

2. Related Work

Implicit neural representation. The novel signal representation paradigm representing a signal as an implicit continuous function by neural networks has gained lots of attention. Recent works have demonstrated the remarkable performance and memory-efficient property in many representation tasks such as 2D image representation [13, 22, 32, 34, 35], occupancy volume representation [7, 24, 30, 36], view synthesis [4, 25, 26, 28, 37, 40] and virtual reality [8]. However, the popular activation function ReLU empirically can't achieve the satisfactory performance with vanilla MLP. Therefore, various modifications have been studied. The mainstream modifications can be classified into two aspects. The first aspect focuses on the input domain. Rahaman *et al*. [31] show that learning high frequency components gets easier with the increasing of the complexity of the input data manifold. Inspired by this phenomenon, Mildenhall *et al*. [26] use the positional encoding which adopts the sinusoidal mapping of the input features as the new input in the view synthesis and achieve remarkable performance. Zhong *et al*. [47] also use this method to reconstruct more accurate continuous distributions of 3D protein structure. Recent studies [19, 23, 38] propose to encode input coordinates by learned features. Inspired by this, Xie *et al*. [42] rearrange the order of input coordinates and obtain more low-frequency components, avoiding confrontation with spectral bias. The second aspect focuses on the activation function. Sitzmann *et al*. [35] find that using periodic functions, such as the sinusoidal function, as activation functions can achieve remarkable performance in signal fitting. Sinusoidal functions are also explored by Fathony *et al*. [13] in multiplicative filter networks. Yüce *et al*. [45] explain the success of the usage of periodic function from a structured dictionary perspective. Inspired by this perspective, Saragadam *et al*. [35] use a complex Gabor wavelet activation and achieve robust and accurate representations.

Spectral bias. The term spectral bias [31] also known as the low-frequency bias [43] implies that MLP tends to learn simple patterns of the real data [2] or the low-frequency components of the target function [43]. Arpit *et al*. [2] first find this phenomenon which has attracted lots of follow up studies. Rahaman *et al*. [31] exploit the structure of ReLU networks to evaluate its Fourier spectrum and estimate the relationships between the spectral norm of MLP weights and the amplitude of the output of MLP at different frequencies. Xu [43] builds a theoretical framework by Fourier Analysis to decompose gradients in the frequency domain and discusses the distributions of the absolute gradients of the parameters at different frequencies. From the perspective of the neural tangent kernel (NTK) theory [18], components of the target function corresponding to larger kernel eigenvalues will be learned faster [5, 33]. Therefore, Tancik *et al*. [39] propose to leverage the eigenvalues of the NTK matrix to analyze the spectral bias of MLP. In this paper, we dive into details of the low-frequency bias issue and find that network reparameterization could provide us a chance to alleviate the bias in network training. We analyze our proposed method with the frequency decomposed loss, gradient [43] and NTK theory [39]. Our experimental results validate our idea of improving spectral bias with weight reparameterization.

Weight reparameterization. In order to reduce the inference cost of deep learning models [15, 17], weight reparameterization method is proposed by Zagoruyko *et al*. [46]. Inspired by this work, various reparameterization methods have been explored to train networks with mergeable auxiliary structures [9–11, 27, 29, 41]. Recently, there are still researches which exploit the idea of reparameterization to design network optimizer for advanced training [12]. In this paper, we adopt the idea of weight reparameterization to improve the approximation accuracy of MLP. For the first time, we link the reparameterization technique with network training bias and theoretically prove the possibility of mitigating frequency-bias with reparameterized training. We hope our theoretical analysis could shed new light on network reparameterization and inspire future studies on advanced reparameterization method.

3. Methodology

3.1. The formulation of INR

The task of INR is to approximate a target function with a multi-layer perceptron (MLP): $f_{\mathbf{\Theta}}(x) \approx g(x)$; where $g(x) : \mathbb{R}^{d_0} \mapsto \mathbb{R}^{d_N}$ is the target function which defines a mapping from a d_0-dimensional real space to a d_N-dimensional real space, and $f_{\mathbf{\Theta}}(x)$ is a N-layer MLP with the learnable parameters set $\mathbf{\Theta}$. Denote the output of n-th layer as:

$$\mathbf{y^{(n)}} = \sigma(\mathbf{W^{(n)}}\mathbf{y^{(n-1)}} + \mathbf{b^{(n)}}), \tag{1}$$

where σ is an element-wise nonlinear activation function; $\mathbf{W^{(n)}} \in \mathbb{R}^{d_n \times d_{n-1}}$ and $\mathbf{b^{(n)}} \in \mathbb{R}^{d_n}$ are the weight and bias for the n-th layer; $\mathbf{y^{(n)}} \in \mathbb{R}^{d_n}$ and $\mathbf{y^{(n-1)}} \in \mathbb{R}^{d_{n-1}}$ are the output and input of the n-th layer, respectively. For $n = N$, we have that $\mathbf{y^{(N)}} = \mathbf{W^{(N)}}\mathbf{y^{(N-1)}} + \mathbf{b^{(N)}}$. The MLP parameters set $\mathbf{\Theta} = \{\mathbf{W^{(i)}}, \mathbf{b^{(i)}}\}_{\mathbf{i=1,\dots,N}}$ is learned by minimizing the loss with gradient-based methods.

The above idea of INR is memory efficient, and the continuous nature of MLP allows INR to model fine detail that is not limited by the grid resolution. However, during the practical optimization process, gradients of parameters are dominated by the error of low-frequency components. Such a spectral bias hinders the accurate learning pace of MLP. Various modifications, including input feature adjustments [39, 42] and activation function adjustments [13, 32, 34, 35], have been exploited for alleviating the spectral bias issue. In this paper, we show that appropriate reparameterization of MLP is also beneficial for narrowing the gap between the gradient magnitude of high-frequency components and low-frequency components.

3.2. Fourier reparameterization

As we have introduced in the previous subsection, the weight matrix in the n-th layer of MLP is denoted as $\mathbf{W^{(n)}} \in \mathbb{R}^{d_n \times d_{n-1}}$. Instead of directly calculating the gradient of $\mathbf{W^{(n)}}$ respect to the loss function, we reparameterize each row of $\mathbf{W^{(n)}}$ as a weighted combination of fixed Fourier bases:

$$\mathbf{W^{(n)}} = \mathbf{\Lambda^{(n)}}\mathbf{B^{(n)}}, \tag{2}$$

where $\mathbf{\Lambda^{(n)}} \in \mathbb{R}^{d_n \times M}$ are the coefficient matrix, and $\mathbf{B^{(n)}} \in \mathbb{R}^{M \times d_{n-1}}$ are M Fourier bases. Each Fourier basis is achieved by changing the frequency ω and phase φ of a cosine function $cos(\omega z + \varphi)$:

$$b_{ij} = cos(\omega_i z_j + \varphi_i), for\ i = 1, \dots, M; j = 1, \dots, d_{n-1}, \tag{3}$$

where $\mathbf{z} = \{z_j\}_{j=1,\dots,d_{n-1}}$ is the sampling position sequence. More details will be introduced in section 5.4.3. Generally, we have $M \geq d_{n-1}$ which means that we reparameterize the weight matrix with over complete bases.

Please note that in Eq. 2, each basis is with the same dimension as the input feature $\mathbf{y^{(n-1)}}$, which means that our reparameterization scheme firstly projects the input features onto a series of Fourier bases and weighted combines the projection coefficients:

$$\mathbf{y^{(n)}} = \sigma(\mathbf{\Lambda^{(n)}}\mathbf{B^{(n)}}\mathbf{y^{(n-1)}} + \mathbf{b^{(n)}}). \tag{4}$$

In the above equation, $\mathbf{B^{(n)}}$ is fixed and we only learn $\mathbf{\Lambda^{(n)}}$ during the training phase. After training, we combine $\mathbf{\Lambda^{(n)}}$ and $\mathbf{B^{(n)}}$ to form the weight matrix $\mathbf{W^{(n)}}$ in the inference. Therefore, our reparameterization approach only adjusts the training dynamic and will not affect the inference process of INR. Moreover, the proposed Fourier reparameterization approach does not affect the input feature space as well as nonlinear activation functions. Our method is compatible with existing techniques, including but not limited to positional encoding [39] and periodic activation functions [35].

3.3. Discussion

In this subsection, we analyze our reparameterization scheme in the frequency domain. By carefully analyzing the gradients of learning parameters respect to different frequencies, we show that appropriate weight reparameterization provides us with a chance to alleviate the low-frequency bias in the network training.

We start our analysis with the definition of some basic concepts [43]. We denote the Fourier Transform of the target function and the MLP representation of INR at frequency k as: $\mathcal{F}[g](k)$ and $\mathcal{F}[f_{\mathbf{\Theta}}](k)$, respectively. Then, the approximation error at frequency k can be naturally achieved by $E(k) = \mathcal{F}[g](k) - \mathcal{F}[f_{\mathbf{\Theta}}](k)$. We further define the following notations: $E(k) = A(k)e^{i\theta(k)}$ and $\mathbb{L}(k) = |E(k)|^2$; where $A(k)$ and $\theta(k) \in [\pi, \pi]$ indicate the amplitude and phase of $E(k)$; $\mathbb{L}(k)$ is the loss component of frequency k; $|\cdot|$ is the norm of the complex number.

Based on the above definitions, Xu [43] shows that the spectral bias can be reflected on the absolute gradient values of parameters at different frequencies:

Theorem 1. *(Theorem 1 in [43]) Consider a MLP with one hidden layer using tanh function $\sigma(x)$ as the activation function. For any frequencies k_1 and k_2 such that $k_1 > k_2 > 0$ and there exist c_1, c_2, such that $A(k_1) > c_1 > 0, A(k_1) < c_2 < \infty$, we have*

$$\lim_{\delta\to 0} \frac{\mu(\{w_j : |\frac{\partial \mathbb{L}(k_2)}{\partial \Theta_{jl}}| > |\frac{\partial \mathbb{L}(k_1)}{\partial \Theta_{jl}}| for\ all\ j,l\} \cap B_\delta)}{\mu(B_\delta)} = 1, \tag{5}$$

where B_δ is a ball with radius δ centered at the origin and $\mu(\cdot)$ is the Lebesgue measure of a set and Θ_{jl} is a leanrable parameter in the parameters set.

Generally, Theorem 1 shows that in the case of one hidden layer MLP, the gradient respect to low-frequency loss $\mathbb{L}(k_2)$ is almost always larger than the one respect to high-frequency part. Although the condition of one hidden layer MLP is not appliable in most of practical cases, based on recent observations of low-frequency bias [2, 18, 31, 43], the above Theorem 1 inspires us to assume such relationship to MLP with more hidden layers. Then, we could have the following Theorem 2:

Theorem 2. *Given a MLP with multiple hidden layers, reparameterize the weight matrix $\mathbf{W} \in \mathbb{R}^{d\times d}$ of one hidden layer with a trainable coefficient matrix $\mathbf{\Lambda} \in \mathbb{R}^{d\times M}$ and the fixed basis matrix $\mathbf{B} \in \mathbb{R}^{M\times d}$. For any frequencies k_1 and k_2 such that $k_1 > k_2 > 0$, given any $\epsilon \geq 0$ and fixed i, for $j = 1, 2, \ldots, M$, there must exist a set of basis matrices such that*

$$|\frac{\partial \mathbb{L}(k_1)}{\partial \lambda_{ij}} / \frac{\partial \mathbb{L}(k_2)}{\partial \lambda_{ij}}| \geq \max\{|\frac{\partial \mathbb{L}(k_1)}{\partial w_{i1}} / \frac{\partial \mathbb{L}(k_2)}{\partial w_{i1}}|, \ldots, |\frac{\partial \mathbb{L}(k_1)}{\partial w_{id}} / \frac{\partial \mathbb{L}(k_2)}{\partial w_{id}}|\} - \epsilon, \tag{6}$$

where $\mathbf{W}(i,j) = w_{ij}$ and $\mathbf{\Lambda}(i,j) = \lambda_{ij}$.

The detailed proof of Theorem 2 can be found in our supplementary file. Theorem 2 implies that reparameterizing MLP weights with appropriate bases is able to enlarge the portion of high-frequency loss components in comparison to low-frequency components, *i.e.* improving the spectral bias in training MLP. Although the optimal basis for frequency-bias adjustment is related to the training data and we are not able to achieve the optimal basis with negligible efforts, we experimentally find that fixed Fourier basis is able to improve the low-frequency bias and provide better INR for various function approximation tasks.

3.4. Implementation details

Basis construction. As we have introduced in section 3.2, we establish Fourier bases with various frequencies and phase parameters. We adopt P different phases and $2F$ different frequencies. Concretely, the φ in Eq. 3 varies from 0 to $2\pi(P-1)/P$ with step length $2\pi/P$; for each phase value, we have a group of low-frequency bases with $\omega = \{1/F, 2/F, \ldots, 1\}$ and a group of high-frequency bases with $\omega = \{1, 2, \ldots, F\}$. Based on the above basis construction scheme, we could obtain $M = 2FP$ bases. Details of the selected F and P values for different settings will be introduced in the experimental section. We also provide ablation experiments in our supplementary field to analyze the effects of different design choices of F and P.

Sampling strategy. The cosine basis function used in Eq. 3 is a continuous function. We need to sample values from the continuous function to achieve discrete basis for weight reparameterization. As the number of sampling points is determined by the neuron number of input feature, the only key hyper-parameter during the sampling process is the range of sampling. To reflect characteristics of different frequency bases, we choose the maximum period ($T_{max} = 2\pi F$) of the adopted Fourier Bases as the sampling range. To maintain the periodicity of bases, uniform sampling is employed. Due to the symmetry of bases, we set the sampling interval as $[-\frac{1}{2}T_{max}, \frac{1}{2}T_{max}]$. The length of interval will be discussed in our ablation studies.

Initialization scheme. Initialization techniques are one of the prerequisites for successfully training a deep neural network. The basic idea of the existing popular initialization strategies [14, 15] is to let the network start in a regime with constant variance between inputs and outputs. While our method reparameterizes the network weights as the fixed Fourier bases and trainable coefficients, initializing the learnable coefficients $\mathbf{\Lambda}$ with existing initialization techniques will deactivate the constant variance requirement. We therefore adjust the initialization strategy for $\mathbf{\Lambda}$. For ReLU activation function [15], we initialize the trainable coefficient matrix $\mathbf{\Lambda^{(n)}}$ using the following equation:

$$\lambda_{ij}^{(n)} \sim U(-\sqrt{\frac{6}{M\sum_{t=1}^{d_{n-1}} {b_{jt}^{(n)}}^2}}, \sqrt{\frac{6}{M\sum_{t=1}^{d_{n-1}} {b_{jt}^{(n)}}^2}}), \tag{7}$$

where $\lambda_{ij}^{(n)}, b_{ij}^{(n)}$ is in the i-th row and j-th column of $\mathbf{\Lambda^{(n)}} \in \mathbb{R}^{d_n\times M}$ and $\mathbf{B^{(n)}} \in \mathbb{R}^{M\times d_{n-1}}$, respectively. For SIREN [35], we have the similar initialization scheme. The detailed derivation can be found in our supplementary file.

4. Experimental Analysis on Simple Function Approximation

We firstly conduct experiments on the simple function approximation task. Thanks to the simplicity of the target function, the behaviour of MLP can be analyzed with various techniques. In the remaining of this section, we firstly

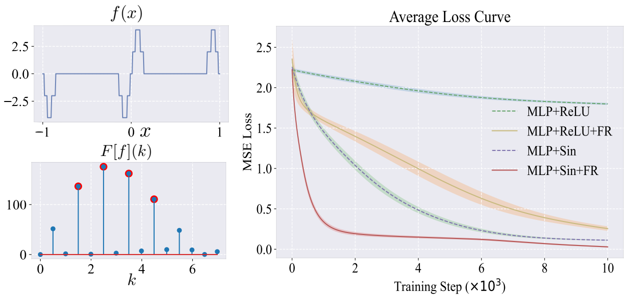

Figure 2. Visualization of simple function. The left side of the first row displays the visualization of the 1D simple function on the $x - y$ coordinate axis. The left side of the second row shows the amplitude of the function in the frequency domain. The right side presents the average loss curve with the shaded area indicating the fluctuation from 100 repetitions.

introduce our experimental settings and then analyze the property of our proposed Fourier reparameterization (FR).

4.1. Experimental settings

In order to thoroughly analyze the advantage of FR, we establish a 1D function $g(x)$ by combining sine functions of different frequencies:

$$2R(\frac{sin(3\pi x)+sin(5\pi x)+sin(7\pi x)+sin(9\pi x)}{2}), \quad (8)$$

where $R(\cdot)$ is the rounding function for increasing the complexity of approximation. A similar rounded periodic function has been adopted in [43] to analyze the spectral-bias of MLP. A visualization of our adopted 1D function and its spectrum can be found in the first column of Fig. 2. As designed by purpose, four distinct peaks marked by red dots can be observed in the stem plot of the spectrum.

We utilize a four-hidden-layer MLP to approximate the $g(x)$ with 300 discrete values uniformly sampled in the interval $[-1, 1]$, where each hidden layer consists of 128 neurons. We conduct experiments on both the ReLU and the periodic activation function Sin. The ω_0 in Sin activation function is set as 5 for the pursuit of fast convergence [35]. For our reparameterization approach, we reparameterize the weight matrices between consecutive hidden layers and set $F = 64$ and $P = 16$. Therefore, we have $M = 2048$ bases in total. The four comparison methods are denoted as: MLP+ReLU, MLP+ReLU+FR, MLP+Sin, MLP+Sin+FR. All the methods are trained with Adam optimizer [21] for 10000 iterations with a learning rate 1e-6 and full-batch.

4.2. Approximation results and analysis

The convergence curves by different methods can be found in Fig. 2. For both the ReLU and Sin activation function cases, Our FR approach improves the convergence speed as well as approximation error of vanilla MLP. Especially for the naive MLP+ReLU case, training network with our proposed reparameterization method could improve the original training paradigm by a large margin.

Frequency-specific error analysis. In [43], using Discrete Fourier Transform, Xu *et al*. compute the relative difference Δ_k between the target signal and the output of MLP at frequency k and empirically show the low-frequency bias of MLP:

$$\Delta_k = \frac{|\mathcal{F}_D[g](k) - \mathcal{F}_D[f_{\Theta}](k)|}{|\mathcal{F}_D[g](k)|}, \quad (9)$$

where $\mathcal{F}_D$ denotes the Discrete Fourier Transform. We follow [43] and use Eq. 9 to analyze the frequency-specific approximation error after different numbers of iterations. As can be found in Fig. 3, the evolution of frequency-specific error clearly shows the low-frequency bias in network training: the error of low-frequency components generally reduces much faster than that of high-frequency components. The proposed FR method narrows the gap in dropping speed of low-frequency error and high-frequency error, thereby leading to overall faster convergence speed.

Neural tangent kernel. Neural tangent kernel (NTK) [18] is a theory to analyze the dynamic training process of neural networks. Tancik *et al*. [39] have shown that components of the target function that correspond to larger kernel eigenvalues will be learned faster and adopt kernel eigenvalues [5, 33, 39] to analyze the spectral bias. Since the conditions of the standard NTK theorey are not applicable to commonly used networks, we follow [1, 45] and utilize the following empirical NTK to analyze the training dynamics of different networks:

$$k'_{NTK}(x_i, x_j) = J_{f_{\Theta}}(x_i) J_{f_{\Theta}}(x_j)^T, \quad (10)$$

where $J_{f_{\Theta}}(x_i)$ denotes the Jacobian matrix of the function f_{Θ} at the i-th sample x_i and $k'_{NTK}(x_i, x_j)$ is the element in the i-th row and j-th column of the empirical NTK matrix.

The first four and the summation of the remaining eigenvalues by different methods are shown in Fig. 4. Consistent with the conclusion of [39], the eigenvalues of MLP + ReLU decay rapidly, which means the model suffers severe spectral bias during training. While, our FR scheme is able to reduce the percentage of the first eigenvalue and enlarge the other eigenvalues of the empirical NTK matrix, leading to more balanced eigenvalue distribution during training.

5. Experimental Results on Vision Applications

Implicit neural representation has been utilized in different vision applications. In this section, we evaluate the proposed FR method on different vision applications.

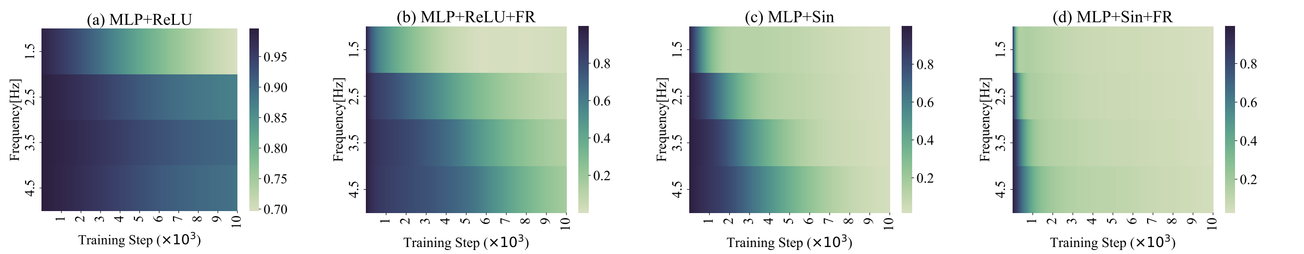

Figure 3. Evolution of frequency-specific approximation error with training iterations of four different methods (x-axis for training step, y-axis for frequency and colormap for relative approximation error).

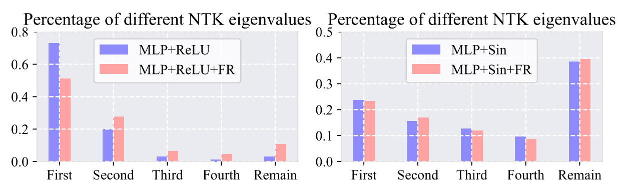

Figure 4. The distribution of different NTK eigenvalues. 'First' denotes the percentage of the largest eigenvalue, and so on. 'Remain' refers to the percentage of the summation of remaining eigenvalues.

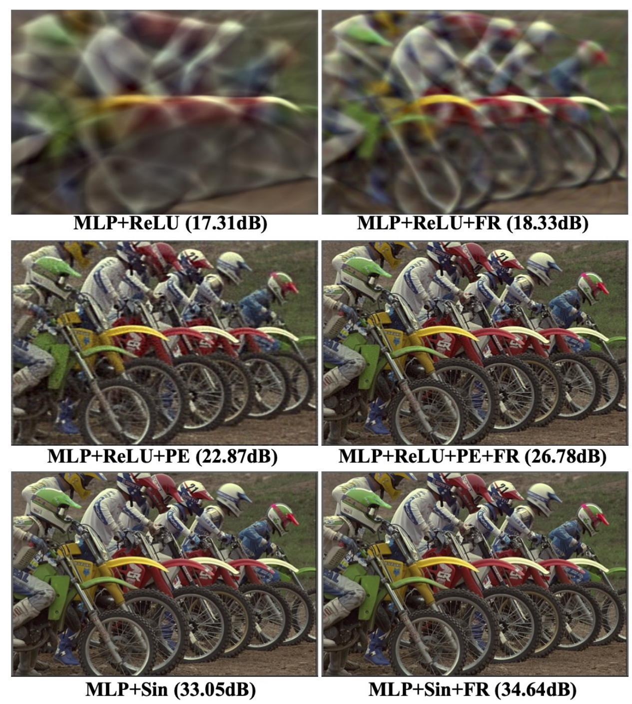

Figure 5. Visual examples of the 2D color image approximation results (PSNR) by different methods. Detailed experimental settings can be found in section 5.1.

5.1. 2D Color image approximation

Natural images are extremely complex functions which simultaneously encompass rich low- and high-frequency components [20]. Single image fitting has become an ideal test bed [34, 35, 42] to evaluate the capability of implicit neural representation. In our experiments, we attempt to parameterize a function $\phi : \mathbb{R}^2 \mapsto \mathbb{R}^3$, $x \mapsto \phi(x)$ that represents a given discrete image in a continuous fashion.

We establish MLP with four hidden layers and each hidden layer contains 256 neurons. We conduct experiments on three MLP architectures, i.e. (1) MLP with RLU as the activation function (MLP+ReLU), (2) MLP + ReLU with Fourier positional encoding (MLP+ReLU+PE) [39], and (3) MLP with Periodic Sin activation function (MLP+Sin) [35]. MLP+ReLU+PE and MLP+Sin represent two important categories of techniques in INR. Experimental results on more activation functions and other input adjustments can be found in our supplementary file. For each MLP architecture, we train baseline model which trains network parameters directly with the standard back-propagation approach and (+FR) model which utilizes our proposed FR scheme in the training phase. We reparameterize the weight matrices between consecutive hidden layers and set $F = 128, P = 32$ for all the images in the experiment. We use Adam optimizer to minimize the ℓ_2 loss between ground truth pixel values and INR approximations. The MLPs are trained with an initial learning rate of 10^{-4} for 3000 iterations, and then we drop the learning rate to 10^{-5} for another 7000 iterations. Full-batch training is adopted.

In Table 1, we report the PSNR values achieved by different INRs for approximating the first 8 images in the Kodak 24 dataset. More evaluation metrics can be found in our supplementary file. Our FR method is able to improve the approximation accuracy for all the three network architectures. Some visual examples of the learned approximations can be found in Fig. 5. Our FR method enables the network to capture more fine details.

5.2. Representing shapes with signed distance functions

Representing shapes with differentiable signed distance functions (SDFs) has the advantage of modeling arbitrary

Table 1. Peak signal to noise ratio (PSNR) of 2D color image approximation results by different methods. Detailed experimental settings can be found in section 5.1.

Method	Kodim 01	Kodim 02	Kodim 03	Kodim 04	Kodim 05	Kodim 06	Kodim 07	Kodim 08	Average
MLP + ReLU	19.37	26.12	25.11	24.57	17.31	21.69	20.79	15.68	21.33
MLP + ReLU + FR	20.34	26.58	27.21	25.72	18.33	22.25	22.47	16.64	22.44
MLP + ReLU + PE	24.47	31.41	31.53	30.16	22.87	26.54	29.33	21.14	27.18
MLP + ReLU + PE + FR	27.64	33.92	34.45	33.23	26.78	29.83	34.13	24.70	30.59
MLP + Sin	31.59	36.55	39.59	36.66	33.05	34.10	39.96	31.00	35.31
MLP + Sin + FR	33.45	38.68	39.58	37.96	34.64	34.45	39.76	32.16	36.34

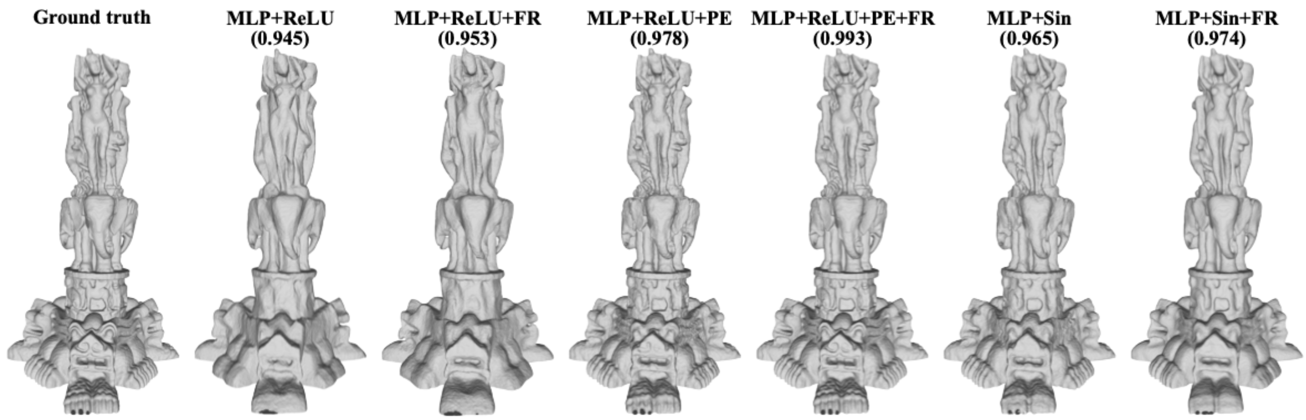

Figure 6. Visual examples of the shape representation results (IOU) by different methods. More experimental details can be found in section 5.2.

topologies [35]. In this section, we evaluate the proposed FR method on the shape representation task. We follow the experimental setting of [34], which sample points over a $512 \times 512 \times 512$ grid. We establish MLP with two hidden layers and each hidden layer contains 256 neurons. We reparameterize the weight matrices between consecutive hidden layers and set $F = 256, P = 8$. We use Adam optimizer to minimize the ℓ_2 loss between sampled voxel values and INR approximations. We train all the networks for 200 epoches with an initial learning rate of 5×10^{-3}. The learning rate is reduced exponentially to 5×10^{-4} .

In Fig. 6, we visualize the shape representation results by different methods, the intersection over union (IOU) metrics are also provided for reference. FR method represents intricate geometric shape with less artifacts and more details.

5.3. Learning neural radiance fields

Learning neural radience fields for view synthesis is also a main application of INR [26, 28, 37]. The main process of the view synthesis task is to reconstruct 3D representation of an object from the given 2D images taken at various given angles. In this section, we evaluate our FR method in this task. The original NeRF and two recent SOTAs by neural networks, *i.e.* the InstantNGP [28] and the DVGO [37], are adopted. We follow the experimental settings of these works and train the models on the all objects of the Blender dataset [26] with 800×800 resolutions. For the original NeRF, we reparameterize the weight matrices of last three hidden layers with $F = 128, P = 32$ and the "NeRF-pytorch" codebase [44] is used. The complete results for the InstantNGP and DVGO can be found in our supplementary file.

In Fig. 7, we show some view synthesis results of the original NeRF. With our proposed FR, the INR is able to capture more details of the complex texture, also representing the more accurate reflection of light.

5.4. Ablation study

In this part, we conduct ablation studies to analyze our design choices. All the ablation experiments are conducted on the first three images of the Kodak 24 dataset, and the structure of the 2D image fitting task is adopted.

5.4.1 Weight reparameterization with Fourier basis

In Theorem 2, we show that appropriate bases selection could alleviate the low-frequency bias issue. In this paper, we select fixed Fourier bases to reparameterize our MLP and show its superiority approximation accuracy on several applications. In this section, we conduct experiments to show that the selection of basis plays a crucial role in our approach. We generate random basis from a uniform distribution and denote the model as random reparameterization (RR). Moreover, we also adopt a similar strategy as

Figure 7. Visual examples of the view synthesis results (PSNR) by learning neural radiance field with the original NeRF [26].

Table 2. Ablation experiments on the reparameterization bases. Our Fourier bases achieve the average apprixmation results (PSNR) on all the three network architectures. Detailed experimental results can be found in Section 5.4.1.

Method	Kodim 01	Kodim 02	Kodim 03	Average
MLP+ReLU+RR	19.64	26.19	25.66	23.83
MLP+ReLU+RIR	20.22	26.48	26.67	24.46
MLP+ReLU+FR	20.34	26.58	27.21	24.71
MLP+ReLU+PE+RR	27.05	32.16	33.38	30.86
MLP+ReLU+PE+RIR	26.91	32.04	32.85	30.60
MLP+ReLU+PE+FR	27.64	33.92	34.45	32.00
MLP+Sin+RR	34.28	37.84	30.48	34.20
MLP+Sin+RIR	25.19	28.61	27.12	26.97
MLP+Sin+FR	33.45	38.68	39.58	37.24

the existing reparameterization works [11] which updates all the random initialized parameters instead of fixed basis during the training phase, and denote the model as randomly initializaed reparameterization (RIR). The average approximation results by different reparameterization schemes are shown in Table 2. The results clearly show that bases play a pivotal role in reparameterized training, and our selected Fourier bases have advantages in 2D image fitting.

5.4.2 Training speed

As shown in Fig. 2, in terms of learning iterations, our FR accelerates the convergence speed of network training. While, since our FR will lead to more computations in each training step, we present the detailed training time by different methods. In Table 3, we show the average per epoch training time by different methods in the simple function approximation experiment. The additional time introduced by our FR is not significant. Although it takes an additional 17.3% time (from 2.89×10^{-3} seconds to 3.31×10^{-3} seconds) for training a MLP + ReLu architecture, our FR could lead to 86% improvement on accuracy.

5.4.3 Sampling interval analysis

Another important hyper-parameter for our method is our sampling interval. In this section, we conduct experiments to analyze the effect of different sampling intervals. Denote the maximum period of the adopted Fourier function as T_{max}. We vary the sampling interval from $0.1T_{max}$ to $10T_{max}$. The approximation accuracy with different sampling intervals can be found in Table 4. Our method could achieve good results with a wide range of sampling intervals, *i.e.* $0.5T_{max}$ to $4T_{max}$. While sampling points from a very small range fails to represent an entire period for many bases and leads to a performance drop.

Table 3. Average per epoch training time by different methods in the simple function approximation experiment. Detaild experimental results can be found in Section 4.1 and Section 5.4.1.

Method	MLP+ReLU	MLP+ReLU+FR	MLP+Sin	MLP+Sin+FR
Time (ms)	2.89	3.31	3.50	3.59

Table 4. Ablation experiments on sampling intervals. Our method could achieve good results (PSNR) on a wide range of sampling intervals. More experimental details can be found in section 5.4.3.

Length ($\times T_{max}$)	0.1	0.25	0.5	1	2	4	10
Average	23.34	24.07	24.38	24.62	24.71	24.53	23.14

6. Conclusions

In this paper, we proposed a novel FR method for advanced INR. We theoretically analyzed the low-frequency bias issue of MLP for INR and show that appropriate network reparameterization is able to alleviate the low-frequency bias in training MLP. Based on our theoretical analysis, we proposed our FR method which learns coefficient matrix of fixed Fourier bases to compose network weights instead of directly learning them from training data. Experiments were conducted on simple function task and real-world vision applications. Our method improved the representation accuracy for a wide range of commonly used INR network architectures. We hope our initial study could inspire future works in adjusting the learning bias of network by advanced network reparameterization.

References

[1] What can linearized neural networks actually say about generalization? *Advances in Neural Information Processing Systems*, 34:8998–9010, 2021. 5

[2] Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. A closer look at memorization in deep networks, 2017. 2, 4

[3] Matan Atzmon and Yaron Lipman. Sal: Sign agnostic learning of shapes from raw data, 2020. 1

[4] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021. 2

[5] Alberto Bietti and Julien Mairal. On the inductive bias of neural tangent kernels. *Advances in Neural Information Processing Systems*, 32, 2019. 2, 5

[6] Rohan Chabra, Jan Eric Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction, 2020. 1

[7] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6970–6981, 2020. 2

[8] Nianchen Deng, Zhenyi He, Jiannan Ye, Budmonde Duinkharjav, Praneeth Chakravarthula, Xubo Yang, and Qi Sun. Fov-nerf: Foveated neural radiance fields for virtual reality. *IEEE Transactions on Visualization and Computer Graphics*, 28(11):3854–3864, 2022. 2

[9] Xiaohan Ding, Yuchen Guo, Guiguang Ding, and Jungong Han. Acnet: Strengthening the kernel skeletons for powerful cnn via asymmetric convolution blocks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1911–1920, 2019. 3

[10] Xiaohan Ding, Tianxiang Hao, Jianchao Tan, Ji Liu, Jungong Han, Yuchen Guo, and Guiguang Ding. Resrep: Lossless cnn pruning via decoupling remembering and forgetting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4510–4520, 2021.

[11] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13733–13742, 2021. 3, 8

[12] Xiaohan Ding, Honghao Chen, Xiangyu Zhang, Kaiqi Huang, Jungong Han, and Guiguang Ding. Re-parameterizing your optimizers rather than architectures. *arXiv preprint arXiv:2205.15242*, 2022. 3

[13] Rizal Fathony, Anit Kumar Sahu, Devin Willmott, and J Zico Kolter. Multiplicative filter networks. In *International Conference on Learning Representations*, 2020. 2, 3

[14] Xavier Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research - Proceedings Track*, 9:249–256, 2010. 4

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3, 4

[16] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991. 1

[17] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 3

[18] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018. 2, 4, 5

[19] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, Thomas Funkhouser, et al. Local implicit grid representations for 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6001–6010, 2020. 2

[20] Ming Jiang. Chan tony f, shen jianhong (jackie):image processing and analysis: variational, pde, wavelet, and stochastic methods,in society for industrial and applied mathematics (siam). *Biomedical Engineering Online - BIOMED ENG ONLINE*, 5:1–3, 2006. 6

[21] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 5

[22] Sylwester Klocek, Łukasz Maziarka, Maciej Wołczyk, Jacek Tabor, Jakub Nowak, and Marek Śmieja. Hypernetwork functional image representation. In *International Conference on Artificial Neural Networks*, pages 496–510. Springer, 2019. 2

[23] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33:15651–15663, 2020. 2

[24] Shi-Lin Liu, Hao-Xiang Guo, Hao Pan, Peng-Shuai Wang, Xin Tong, and Yang Liu. Deep implicit moving least-squares functions for 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1788–1797, 2021. 2

[25] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7210–7219, 2021. 2

[26] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020. 1, 2, 7, 8

[27] Hesham Mostafa and Xin Wang. Parameter efficient training of deep convolutional neural networks by dynamic sparse

reparameterization. In *International Conference on Machine Learning*, pages 4646–4655. PMLR, 2019. 3

[28] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, 2022. 1, 2, 7

[29] Jane Park and Young Hoon Kwak. Design-bid-build (dbb) vs. design-build (db) in the us public transportation projects: The choice and consequences. *International Journal of Project Management*, 35(3):280–295, 2017. 3

[30] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation, 2019. 1, 2

[31] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A. Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks, 2019. 1, 2, 4

[32] Sameera Ramasinghe and Simon Lucey. Beyond periodicity: Towards a unifying framework for activations in coordinate-mlps. In *European Conference on Computer Vision*, pages 142–158. Springer, 2022. 2, 3

[33] Basri Ronen, David Jacobs, Yoni Kasten, and Shira Kritchman. The convergence rate of neural networks for learned functions of different frequencies. *Advances in Neural Information Processing Systems*, 32, 2019. 2, 5

[34] Vishwanath Saragadam, Daniel LeJeune, Jasper Tan, Guha Balakrishnan, Ashok Veeraraghavan, and Richard G. Baraniuk. Wire: Wavelet implicit neural representations, 2023. 1, 2, 3, 6, 7

[35] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions, 2020. 1, 2, 3, 4, 5, 6, 7

[36] Edward Smith, David Meger, Luis Pineda, Roberto Calandra, Jitendra Malik, Adriana Romero Soriano, and Michal Drozdzal. Active 3d shape reconstruction from vision and touch. *Advances in Neural Information Processing Systems*, 34:16064–16078, 2021. 2

[37] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 2, 7

[38] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural geometric level of detail: Real-time rendering with implicit 3d shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11358–11367, 2021. 2

[39] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains, 2020. 2, 3, 5, 6

[40] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T Barron, and Pratul P Srinivasan. Ref-nerf: Structured view-dependent appearance for neural radiance fields. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5481–5490. IEEE, 2022. 2

[41] Qi Xie, Qian Zhao, Zongben Xu, and Deyu Meng. Fourier series expansion based filter parametrization for equivariant convolutions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4537–4551, 2022. 3

[42] Shaowen Xie, Hao Zhu, Zhen Liu, Qi Zhang, You Zhou, Xun Cao, and Zhan Ma. Diner: Disorder-invariant implicit neural representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6143–6152, 2023. 2, 3, 6

[43] Zhiqin John Xu. Understanding training and generalization in deep learning by fourier analysis, 2018. 1, 2, 3, 4, 5

[44] Lin Yen-Chen. Nerf-pytorch. `https://github.com/yenchenlin/nerf-pytorch/`, 2020. 7

[45] Gizem Yüce, Guillermo Ortiz-Jiménez, Beril Besbinar, and Pascal Frossard. A structured dictionary perspective on implicit neural representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19228–19238, 2022. 2, 5

[46] Sergey Zagoruyko and Nikos Komodakis. Diracnets: Training very deep neural networks without skip-connections. *arXiv preprint arXiv:1706.00388*, 2017. 3

[47] Ellen D Zhong, Tristan Bepler, Joseph H Davis, and Bonnie Berger. Reconstructing continuous distributions of 3d protein structure from cryo-em images. *arXiv preprint arXiv:1909.05215*, 2019. 2