Content-Aware Frequency Encoding for Implicit Neural Representations with Fourier-Chebyshev Features

Supplementary Material

We provide the proof of Theorem 3 in Section A. The parameter settings for the main experiments in the paper are described in Section B. Additional experiments are presented in Section C, and further ablation studies are included in Section D.

A. Proofs of Theorems

Before presenting the theorem, we first restate the framework of INRs. INRs are typically defined as a coordinate mapping function $\gamma : \mathbb{R}^D \to \mathbb{R}^M$ followed by an MLP. The MLP is parameterized by weight matrices $W^{(\ell)} \in \mathbb{R}^{F_{\ell-1} \times F_\ell}$, bias vectors $b^{(\ell)} \in \mathbb{R}^{F_\ell}$, and element-wise activation functions $\rho^{(\ell)} : \mathbb{R} \to \mathbb{R}$, applied at each layer $\ell = 1, \ldots, L-1$. Let $z^{(\ell)}$ denote the post-activation output of layer ℓ. The forward computation is expressed as:

$$\begin{aligned} z^{(0)} &= \gamma(x), \\ z^{(\ell)} &= \rho^{(\ell)}\left(W^{(\ell)} z^{(\ell-1)} + b^{(\ell)}\right), \quad \ell = 1, \ldots, L-1, \\ f_{\boldsymbol{\theta}}(x) &= W^{(L)} z^{(L-1)} + b^{(L)}. \end{aligned} \tag{9}$$

Theorem. *Let $f_{\boldsymbol{\theta}} : \mathbb{R} \to \mathbb{R}$ be an INR of the form of Eq. (9) with depth L, where the activation functions are polynomials of degree at most K, i.e., $\rho(z) = \sum_{k=0}^{K} \alpha_k z^k$. Let $J = \{j_1, \ldots, j_M\} \subset \mathbb{N}$ be a set of M arbitrary non-negative integer orders. The input mapping $\gamma(x)$ embeds the coordinate x into a vector of Chebyshev polynomials:*

$$z^{(0)} = \gamma(x) = \left[T_{j_1}(x), T_{j_2}(x), \ldots, T_{j_M}(x)\right]^\top.$$

Specifically, the functional space of this architecture is constrained to the form:

$$f_{\boldsymbol{\theta}}(x) = \sum_{m \in \mathcal{H}(J)} c_m T_m(x),$$

where $c_m \in \mathbb{R}$ are coefficients determined by the network parameters, and the set of reachable spectral orders $\mathcal{H}(J)$ is a subset of the integer combination set:

$$\mathcal{H}(J) \subseteq \tilde{\mathcal{H}}(J) := \Big\{ h = |\sum_{t=1}^{M} s_t j_t| \;\Big|\; s_t \in \mathbb{Z}, \sum_{t=1}^{M} |s_t| \le K^{L-1} \Big\}.$$

Since Chebyshev polynomials share the trigonometric properties of cosine functions, they satisfy the product-to-sum identity:

$$T_p(x) T_q(x) = \tfrac{1}{2}\left[\, T_{p+q}(x) + T_{|p-q|}(x) \,\right]. \tag{10}$$

Leveraging this property, we extend the theoretical framework to Chebyshev polynomials via the following lemmas.

Lemma 1 (Chebyshev Product)**.** *Let $\{T_p(x)\}_{p \in P}$ and $\{T_q(x)\}_{q \in Q}$ be two collections of first-kind Chebyshev polynomials indexed by sets $P, Q \subset \mathbb{N}$. Let $\{\beta_p^{(1)}\}_{p \in P}$ and $\{\beta_q^{(2)}\}_{q \in Q}$ be their corresponding scalar coefficients. Then,*

$$\left(\sum_{p \in P} \beta_p^{(1)} T_p(x)\right)\left(\sum_{q \in Q} \beta_q^{(2)} T_q(x)\right) = \sum_{h \in \mathcal{S}} \hat{\beta}_h T_h(x), \tag{11}$$

where the resulting index set $\mathcal{S}$ is defined as:

$$\mathcal{S} = \left\{\, h \mid h = p+q \text{ or } h = |p-q|,\ p \in P,\ q \in Q \,\right\}, \tag{12}$$

and $\{\hat{\beta}_h \mid h \in \mathcal{S}\} \subset \mathbb{R}$ are the combined coefficients.

Proof. Expanding the product of the two sums and applying the identity in Eq. (10):

$$\begin{aligned} &\left(\sum_{p \in P} \beta_p^{(1)} T_p(x)\right)\left(\sum_{q \in Q} \beta_q^{(2)} T_q(x)\right) \\ &= \sum_{p \in P} \sum_{q \in Q} \beta_p^{(1)} \beta_q^{(2)} T_p(x) T_q(x) \\ &= \sum_{p \in P} \sum_{q \in Q} \frac{1}{2} \beta_p^{(1)} \beta_q^{(2)} \left[\, T_{p+q}(x) + T_{|p-q|}(x) \,\right] \\ &= \sum_{h \in \mathcal{S}} \hat{\beta}_h T_h(x). \end{aligned} \tag{13}$$

The final step follows by grouping terms associated with the same Chebyshev order $h \in \mathcal{S}$. □

Lemma 2 (Chebyshev Power)**.** *Let $\{T_j(x)\}_{j \in J}$ be a collection of first-kind Chebyshev polynomials indexed by $J \subset \mathbb{N}$, with coefficients $\{\beta_j\}_{j \in J} \subset \mathbb{R}$. For any integer $k \ge 1$,*

$$\left(\sum_{j \in J} \beta_j T_j(x)\right)^k = \sum_{h \in \mathcal{S}_k} \tilde{\beta}_h T_h(x), \tag{14}$$

where the order set $\mathcal{S}_k$ is recursively defined as:

$$\begin{aligned}\mathcal{S}_1 &= J,\\ \mathcal{S}_{k+1} &= \mathcal{D}(\mathcal{S}_k, J) := \big\{\, h' \mid h' = h + j \text{ or } \\ &\qquad h' = |h - j|,\ h \in \mathcal{S}_k,\ j \in J \,\big\}.\end{aligned}$$

Moreover, $\mathcal{S}_k$ is a subset of the "order combination" set $\tilde{\mathcal{S}}_k(J)$:

$$\mathcal{S}_k \subseteq \tilde{\mathcal{S}}_k(J) := \Big\{ h = \Big|\sum_{j\in J} c_j j\Big| \;\Big|\; c_j \in \mathbb{Z},\ \sum_{j\in J}|c_j| \le k \Big\}. \tag{15}$$

Proof. We proceed by induction.

1. Proof of Eq. (14) (Recursive Structure)

Base Case ($k = 1$): Trivially, $\mathcal{S}_1 = J$.

Inductive Step: Assume Eq. (14) holds for some $k \ge 1$. Then,

$$\begin{aligned}\left(\sum_{j\in J}\beta_j T_j(x)\right)^{k+1} &= \left(\sum_{j\in J}\beta_j T_j(x)\right)^{k}\left(\sum_{j\in J}\beta_j T_j(x)\right)\\ &= \left(\sum_{h\in\mathcal{S}_k}\tilde{\beta}_h T_h(x)\right)\left(\sum_{j\in J}\beta_j T_j(x)\right).\end{aligned} \tag{16}$$

Applying the Chebyshev Product Lemma to Eq. (16), the resulting indices h' belong to the set formed by the sum and difference of indices in $\mathcal{S}_k$ and J. Thus, $\mathcal{S}_{k+1} = \mathcal{D}(\mathcal{S}_k, J)$, and Eq. (14) holds for $k+1$.

2. Proof of $\mathcal{S}_k \subseteq \tilde{\mathcal{S}}_k(J)$

Base Case ($k = 1$): Any $j \in J$ can be written as $|\sum_{i\in J} c_i i|$ with $c_j = 1$ and others zero, satisfying $\sum|c_i| = 1$. Thus $\mathcal{S}_1 \subseteq \tilde{\mathcal{S}}_1(J)$.

Inductive Step: Assume $\mathcal{S}_k \subseteq \tilde{\mathcal{S}}_k(J)$. Consider $h' \in \mathcal{S}_{k+1}$. By definition, $h' = |h \pm j|$ for some $h \in \mathcal{S}_k$ and $j \in J$. By the inductive hypothesis, $h = |\sum_{i\in J} c_i i|$ with $\sum|c_i| \le k$. Using $T_{-n}(x) = T_n(x)$, we can absorb signs into the coefficients:

$$h' = \Big|\Big(\pm\sum_{i\in J} c_i i\Big) \pm j\Big| = \Big|\sum_{i\in J} c'_i i\Big|, \tag{17}$$

where $c'_j = \pm c_j \pm 1$ and $c'_i = \pm c_i$ for $i \ne j$. The complexity satisfies $\sum|c'_i| \le \sum|c_i| + 1 \le k + 1$. Therefore, $h' \in \tilde{\mathcal{S}}_{k+1}(J)$. □

With the above lemmas established, we now proceed to prove the theorem.

Proof of Theorem. The proof proceeds by induction on the layer index ℓ to track the propagation of the order complexity.

Base Case ($\ell = 0$). The initial representation is $z^{(0)} = [T_{j_1}(x), \dots, T_{j_M}(x)]^\top$. Each component corresponds to a Chebyshev polynomial of order $m \in J = \{j_1, \dots, j_M\}$. Evaluating the coefficient complexity defined in $\tilde{\mathcal{H}}(J)$, for the t-th component $T_{j_t}(x)$, we can define the integer weights as $s_t = 1$ and $s_{t'} = 0$ for $t' \ne t$. The resulting spectral complexity is $\sum_{i=1}^M |s_i| = 1 = K^0$. Thus, the statement holds for $\ell = 0$.

Inductive Hypothesis. Assume that for the hidden layer $\ell - 1$ (where $1 \le \ell < L$), every component of $z^{(\ell-1)}$ is a linear combination of Chebyshev polynomials $T_m(x)$, where each order m belongs to the set:

$$\Omega_{\ell-1} = \Big\{ \Big|\sum_{t=1}^M s_t j_t\Big| \;\Big|\; \sum_{t=1}^M |s_t| \le K^{\ell-1} \Big\}.$$

Inductive Step. Consider the representation at layer ℓ:

$$z^{(\ell)} = \rho^{(\ell)}\left(W^{(\ell)} z^{(\ell-1)} + b^{(\ell)}\right).$$

Let $v = W^{(\ell)} z^{(\ell-1)} + b^{(\ell)}$ be the pre-activation vector. Since $z^{(\ell-1)}$ consists of polynomials with orders in $\Omega_{\ell-1}$, and the bias $b^{(\ell)}$ corresponds to $T_0(x)$ (where all $s_t = 0$, satisfying $\sum|s_t| = 0 \le K^{\ell-1}$), every component of v is a linear combination of Chebyshev polynomials with orders in $\Omega_{\ell-1}$.

The activation function is a polynomial of degree K, i.e., $\rho(z) = \sum_{k=0}^K \alpha_k z^k$. We analyze the term v^k for any component of v. By the Lemma 2, raising a sum of Chebyshev polynomials to the k-th power generates new orders. Specifically, if the input orders have coefficient complexity bounded by C, the output orders have complexity bounded by $k \cdot C$.

Applying this to our case with input complexity $C = K^{\ell-1}$ and maximum power $k = K$, the resulting orders m in $z^{(\ell)}$ satisfy:

$$\sum_{t=1}^M |s'_t| \le K \cdot K^{\ell-1} = K^\ell.$$

Thus, the elements of $z^{(\ell)}$ are composed of Chebyshev polynomials with orders in Ω_ℓ.

Finally, the network output is a linear transformation $f_{\boldsymbol{\theta}}(x) = W^{(L)} z^{(L-1)} + b^{(L)}$. Since the linear operation does not increase the polynomial degree complexity, the spectral orders remain bounded by that of $z^{(L-1)}$. Substituting $\ell = L - 1$ into our inductive result, the output orders belong to $\mathcal{H}(J)$ with the constraint:

$$\sum_{t=1}^M |s_t| \le K^{L-1}.$$

This concludes the proof. □

B. Experimental Settings for Main Tasks

We briefly review the CAFE+ and clarify the two symbols M and J used in our experimental descriptions. Given a coordinate input $\mathbf{x} \in \mathbb{R}^D$, CAFE+ maps it to a high-dimensional vector through

$$\Psi(\mathbf{x}) = \bigodot_{i=1}^{N} \left\{ \mathbf{W}_i \left[\Phi_{\mathrm{FF}}(\mathbf{x}),\ \Phi_{\mathrm{CF}}(\mathbf{x}) \right] + \mathbf{b}_i \right\}, \tag{18}$$

where Φ_{FF} and Φ_{CF} denote the Fourier and Chebyshev encodings, respectively.

The Fourier feature mapping is defined as

$$\Phi_{\mathrm{FF}}(\mathbf{x}) = \left[\sin(2\pi \boldsymbol{\omega}_i^\top \mathbf{x}),\ \cos(2\pi \boldsymbol{\omega}_i^\top \mathbf{x}) \right]_{i=1}^{M}, \tag{19}$$

where M indicates the number of sampled Fourier frequencies.

The Chebyshev feature mapping is given by

$$\Phi_{\mathrm{CF}}(\mathbf{x}) = \left[T_j(x_d) \right]_{d=1,\ldots,D;\, j=0,\ldots,J-1} \in \mathbb{R}^{DJ}, \tag{20}$$

where $T_j(\cdot)$ is the Chebyshev polynomial of degree j, and J specifies the number of Chebyshev basis functions per dimension.

Throughout our experiments, we use M and J to explicitly denote the number of Fourier features and Chebyshev features employed in the coordinate encoding.

Implementation Note. For our method, the number of Fourier features M and the number of Chebyshev features J need to be specified. In practice, our model is relatively robust to the choice of J, and we recommend choosing M and J such that the feature dimensions after Fourier and Chebyshev encoding maintain an approximate ratio of $3:1$. This ratio does not need to be strictly satisfied; it simply provides a convenient guideline for selecting M and J. For example, in a 2D image fitting task with a target hidden dimension of 256, one may set $M = 96$ and $J = 32$. After encoding, the Fourier features produce a vector of length $2M = 192$, while the Chebyshev features yield a vector of length $2J = 64$. Their ratio is therefore $192 : 64 = 3 : 1$, and the concatenated representation matches the desired dimension of 256.

We provide the detailed experimental settings for 2D image fitting, 3D shape representation, and NeRF experiments as follows:

- **2D Image Fitting.** For SIREN [37] and FINER [21], we use $\omega_0 = 30$ and three hidden layers. For Wire [35], we use $\omega_0 = 20$, $s_0 = 30$ and two hidden layers. SCONE [18] uses three hidden layers with $\omega_0 = 30$ and $\omega_\ell = [90, 60, 30, 10]$. SL^{2}A [33] uses three hidden layers with degree $= 512$ and rank $= 128$. Our method employs three parallel linear layers with an RFF scale of 30. The default configuration uses Fourier features $M = 88$ and Chebyshev features $J = 30$ with one hidden layer in the backbone MLP, while the larger-parameter variant adopts $M = 96$ and $J = 32$ with two backbone MLP hidden layers. Except for SCONE which uses 250 hidden neurons, all other methods use 256 hidden neurons. We train all models for 6,000 iterations.
- **3D Shape Representation.** For SIREN and FINER, we use $\omega_0 = 30$ and three hidden layers. Wire uses $\omega_0 = 20$, $s_0 = 30$ and two hidden layers. SCONE uses two hidden layers with $\omega_0 = 30$ and $\omega_\ell = [60, 30, 10]$. SL^{2}A uses two hidden layers with degree $= 256$ and rank $= 128$. Our method adopts two parallel linear layers with an RFF scale of 30. We use $M = 98$ Fourier features and $J = 20$ Chebyshev features, ensuring that the encoded dimension is $2M + 3J = 256$. Except for SCONE which uses 250 hidden neurons, all other methods use 256 hidden neurons. During training, all voxels are randomly shuffled in each iteration and processed in mini-batches of up to 200,000 points, for a total of 200 iterations.
- **NeRF [26].** Our implementation follows the settings provided by FINER. For our method, we set the frequency features in PE to 10 and concatenate Chebyshev features with $J = 8$, resulting in a concatenated feature length of $3 \times 2 \times 10 + 3 \times 8 = 84$. Two parallel linear projections map these features to length 182 and are combined via Hadamard product. The backbone MLP has two hidden layers with 182 neurons each. This produces a 182-dimensional feature vector. Excluding the first feature, the remaining 181 features are concatenated with the input coordinates, processed through two parallel linear layers, and then passed through the two-layer backbone MLP to obtain the final output.

Additional visualizations are provided for 2D image fitting (Fig. 19), 3D shape representation (Fig. 20), and NeRF (Fig. 21).

C. Additional Experiments

C.1. Full-Resolution DIV2K Representation

We perform image fitting on DIV2K [1] images 0873 (2040×1456), 0878 (2040×1740), and 0891 (1668×2040) at full resolution, as well as their 2× downsampled versions. We train for 6,000 iterations, randomly sampling 512×512 points per iteration. All methods use hidden layers with 384 neurons. For SIREN and FINER, we adopt three hidden layers with $\omega_0 = 30$. For Wire, we use two hidden layers with $\omega_0 = 20$ and $s_0 = 20$. For SCONE, we use three hidden layers with $\omega_0 = 30$ and $\omega_\ell = [90, 60, 30, 10]$. For SL^{2}A, we use three hidden layers with degree $= 512$ and rank $= 192$. For our method, we employ three parallel linear layers with an RFF scale of 30, using $M = 144$ Fourier features and $J = 48$ Chebyshev features. We present the

Table 6. PSNR comparison on DIV2K images 0873, 0878, and 0891 under full-resolution and $\times 2$ downsampling settings. The number of parameters and average inference time per image are also reported.

Method	0873	0873 ($\times 2$)	0878	0878 ($\times 2$)	0891	0891 ($\times 2$)	Params (M)	Avg. Time (s)
SIREN	24.91	31.56	32.42	36.94	27.20	32.48	0.45	292.50
WIRE	23.45	28.98	29.12	32.04	23.99	27.23	0.18	707.48
SCONE	<u>26.10</u>	33.14	<u>34.09</u>	<u>39.12</u>	<u>27.68</u>	<u>33.19</u>	0.45	647.06
FINER	25.28	31.59	32.66	36.86	26.77	32.07	0.45	375.28
SL^{2}A	25.95	<u>34.28</u>	34.00	38.59	26.82	32.53	0.84	456.12
Ours	**29.65**	**37.18**	**35.79**	**40.82**	**30.79**	**36.46**	0.74	326.95

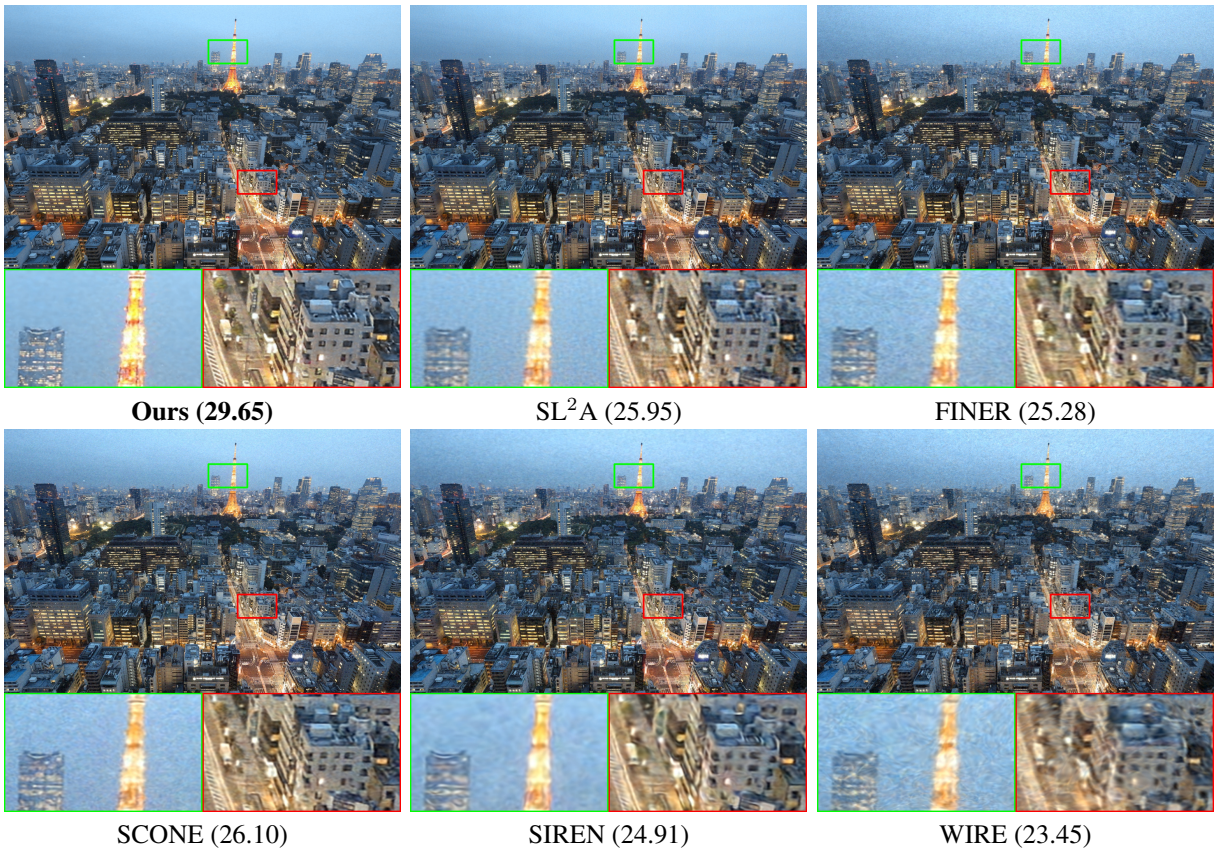


Figure 11. Visualization results on the full-resolution DIV2K image 0873, where we zoom in on two selected regions.

visualization results of the DIV2K image 0873 at its original resolution in Fig. 11. The PSNR, model parameters, and training time are reported in Table 6. These results further demonstrate the effectiveness of our method.

C.2. Image Restoration

We applied our framework to three classical image restoration tasks, namely image denoising, inpainting, and super-resolution. Specifically, we considered Gaussian noise with a standard deviation of 0.2, randomly removed 50% of pixels for inpainting, and performed $\times 4$ super-resolution.

The detailed configurations for each task are summarized below:

- **Denoising.** All methods use hidden layers with 256 neurons and two hidden layers. SIREN and FINER use $\omega_0 = 10$; Wire uses $\omega_0 = 4$ and $s_0 = 4$; SCONE uses $\omega_0 = 2$ with $\omega_\ell = [60, 30, 10]$; SL^{2}A uses degree $= 64$ and rank $= 64$. For our method, we use two parallel linear layers with $M = 96$ Fourier features, $J = 32$ Chebyshev features, and an RFF scale of 2.
- **Inpainting.** All methods use hidden layers with 256 neurons and two hidden layers. SIREN and FINER use

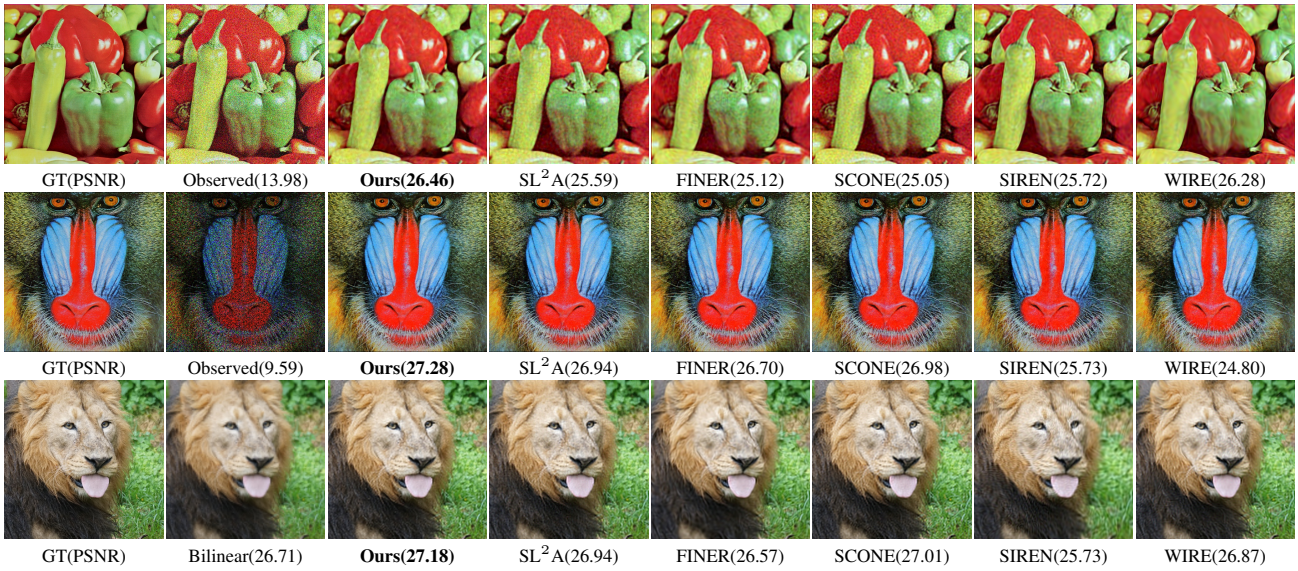
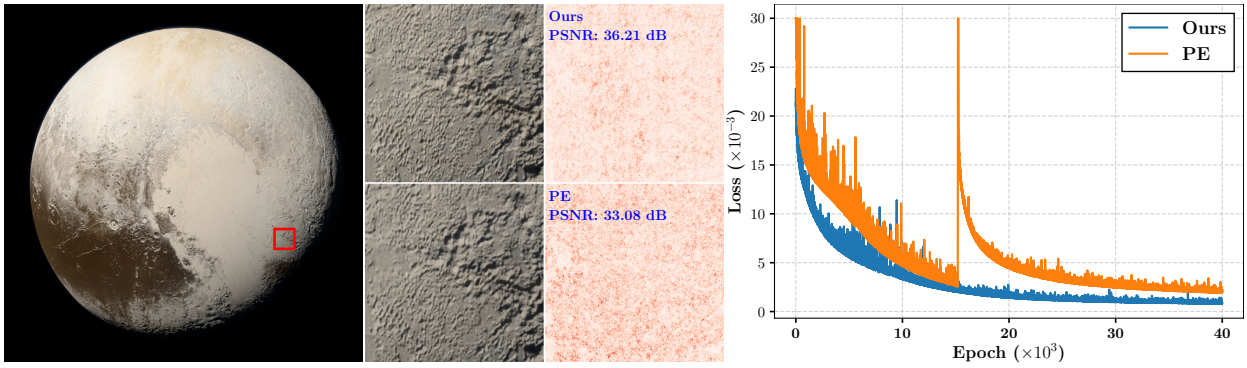


Figure 12. Visualization of image denoising, inpainting, and single-image super-resolution tasks (top to bottom). For denoising, Gaussian noise with standard deviation 0.2 is added. Inpainting randomly removes 50% of pixels. Super-resolution uses a scale factor of 4.



(a) GT (b) Cropped patch with residual (c) Training loss curve

Figure 13. Visualization of the gigapixel Pluto image fitting results. Subfigure (a) shows the original 8000×8000 image, (b) presents the 512×512 patches fitted by our method and PE, along with the corresponding residual maps (c) displays the training loss curve.

$\omega_0 = 30$; Wire uses $\omega_0 = 20$ and $s_0 = 20$; SCONE uses $\omega_0 = 30$ with $\omega_\ell = [60, 30, 10]$; SL^{2}A uses degree $= 256$ and rank $= 128$. For our method, we use two parallel linear layers with $M = 96$ Fourier features, $J = 32$ Chebyshev features, and an RFF scale of 30.

- **Super-resolution** ($\times 4$)**.** All methods use hidden layers with 256 neurons and two hidden layers. SIREN and FINER use $\omega_0 = 10$; Wire uses $\omega_0 = 8$ and $s_0 = 6$; SCONE uses $\omega_0 = 2$ with $\omega_\ell = [60, 30, 10]$; SL^{2}A uses degree $= 64$ and rank $= 64$. For our method, we use two parallel linear layers with $M = 96$ Fourier features, $J = 32$ Chebyshev features, and an RFF scale of 2.

For each task, we select one representative image from standard datasets and visualize the results in Fig. 12. Our method also demonstrates strong performance across these image restoration tasks.

C.3. Gigapixel Image Representation

We further demonstrate the superiority of the proposed CAFE+ on a gigapixel image representation task. Following ACORN [24], we use a widely adopted 8000×8000 Pluto image and train for 40k iterations. For the baseline, the PE method employs an MLP with four hidden layers of 1,536 neurons each. In contrast, our method augments the original PE encoding by concatenating Chebyshev features with $J = 8$, followed by two linear layers (each mapping 62 inputs to 1,536 outputs) whose outputs are combined via a Hadamard product before being fed into the backbone MLP. This modification allows us to use a backbone with only two hidden layers of 1,536 neurons while keeping all other set-

tings identical to PE. As a result, the number of parameters is reduced from 9.52M to 4.92M. As shown in Fig. 13, our method reconstructs finer details, achieves a 3.13 dB PSNR improvement, and exhibits faster and more stable convergence. These results clearly demonstrate the effectiveness of our approach.

C.4. Larger Scale NeRF and Scene-Level SDF Reconstruction

Larger Scale NeRF. We further evaluate NeRF using all training images at the original resolution on the standard scenes (*Lego* / *Ship* / *Drums* / *Hotdog*). We compare our method with the two strongest baselines, SIREN and FINER. The quantitative results are shown in Table 7. **Scene-Level SDF.** We evaluate the methods on the *Replica room0* scene using IoU as the metric under the same parameter budget. The results are: SIREN (0.854), FINER (0.844), and Ours (**0.899**).

Table 7. Quantitative comparison on NeRF scenes using all training images at the original resolution. PSNR is reported in dB.

Method	Lego	Ship	Drums	Hotdog
SIREN	26.94	20.62	23.09	30.89
FINER	28.00	20.88	**23.65**	31.64
Ours	**30.26**	**21.92**	23.51	**32.35**

C.5. Comparison with MFN and BACON

MFN, BACON, and our method all employ the Hadamard product, where MFN and BACON progressively synthesize frequency components across layers via a serial recursive architecture. In contrast, our approach is encoding-based and adopts a parallel design, which leads to faster training and removes the need for carefully modulated initialization. As summarized in Table 8, our method achieves consistently better reconstruction quality while significantly reducing training time compared with MFN and BACON.

Table 8. Comparison with BACON and MFN on Image Fitting

Method	D2K0	D2K1	D2K2	D2K3	D2K4	D2K5	D2K6	D2K7	Params	Time (s)
MFN	35.6	40.1	36.4	41.9	38.4	35.2	39.9	38.9	0.33M	279.1
BACON	34.0	39.0	37.6	39.6	39.5	35.7	38.2	36.0	0.33M	245.6
Ours	**39.5**	**44.3**	**44.2**	**44.7**	**45.0**	**39.2**	**42.6**	**45.0**	0.33M	**149.8**

D. Additional Ablation Studies

D.1. Comparison of Chebyshev and Fourier Features on 1D Function Fitting

In Fig. 14, we compare Fourier features and Chebyshev features (denoted as CF in the figure) by fitting two 1D functions to intuitively reveal their respective strengths. In our experimental setup, each coordinate is mapped to a 32-dimensional feature vector, and the RFF scale is set to 5.

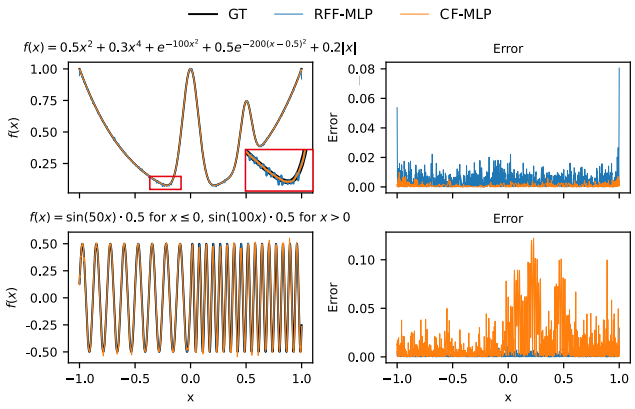
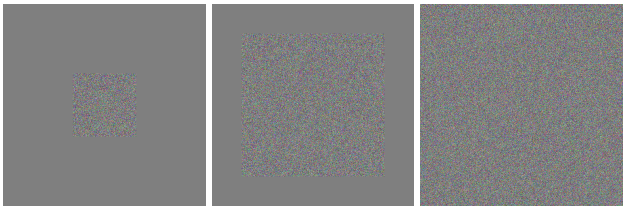


Figure 14. Comparison of RFF and CF in function fitting. CF excels at capturing smooth, low-frequency components but struggles with high-frequency variations, whereas RFF effectively represents high-frequency content yet introduces oscillatory errors on low-frequency functions.



(a) ratio=0.1 (b) ratio=0.5 (c) ratio=1.0

Figure 15. Visualization of the constructed noisy images under different noise ratios.

The MLP architecture consists of a single hidden layer with 32 neurons. We sample 1,000 points for training over 2,000 epochs and evaluate the fitting error on 4,000 uniformly sampled test points. As shown in the figure, Chebyshev features exhibit clear advantages when modeling smooth function regions, whereas Fourier features tend to introduce oscillations that lead to noisy predictions. Conversely, when approximating high-frequency functions, Chebyshev features incur significantly larger errors, while Fourier features maintain superior accuracy.

D.2. Impact of High-Frequency Ratios on Reconstruction

We introduce Chebyshev features with the primary motivation of enhancing the representation of low-frequency signals. However, this naturally raises an important concern regarding whether incorporating Chebyshev features may affect the fitting capability of CAFE when the signal contains a large proportion of high-frequency components.

To investigate this, we construct a synthetic dataset specifically designed to control the ratio of high-frequency content. As shown in Fig. 15, the dataset consists of a high-frequency central region and a flat background. Let $\mathbf{I}_{\text{noise}}$

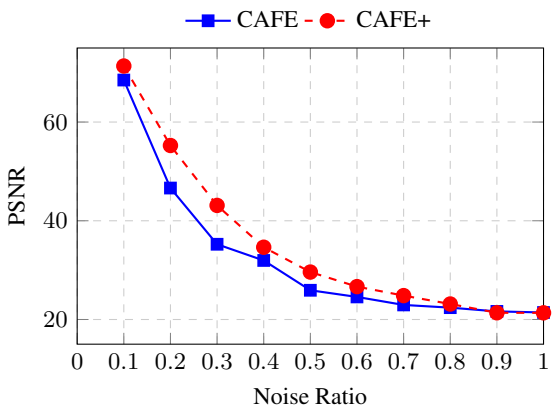


Figure 16. PSNR comparison of CAFE and CAFE+ across different noise ratios.

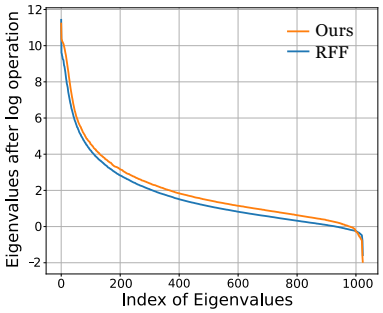


Figure 17. NTK eigenvalues visualization.

denote a Gaussian noise map and $\mathbf{I}_{\text{smooth}} = \mathbf{0}$ represent a zero-filled background. We define a central binary mask $\mathbf{O}\rho$, where the parameter $\rho \in (0, 1]$ determines the ratio of the masked area to the total image area. The final ground-truth image $\mathbf{I}$ is synthesized as

$$\mathbf{I} = \mathbf{O}\rho \odot \mathbf{I}_{\text{noise}} + (1 - \mathbf{O}\rho) \odot \mathbf{I}_{\text{smooth}},$$

where $\odot$ denotes element-wise multiplication. In our experiments, we vary ρ from 0.1 to 1.0 to simulate different proportions of high-frequency information.

As illustrated in the comparison between CAFE+ and CAFE, CAFE+ consistently outperforms CAFE across a wide range of ratios. When the ratio exceeds 0.9, its performance becomes comparable to that of CAFE. This indicates that CAFE+ effectively enhances the representation of mixed spectral distributions without compromising performance in high-frequency dominant scenarios, demonstrating the practical applicability and robustness of our method.

D.3. NTK Eigenvalues

We visualize the NTK eigenvalue of CAFE and RFF, as shown in Fig. 17. CAFE exhibits a more favorable eigenvalue distribution.

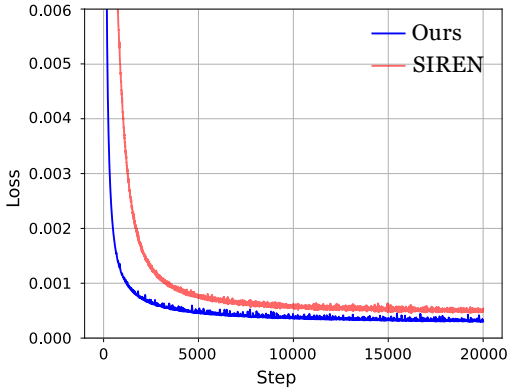


Figure 18. Training loss curves of our method and SIREN. Our method converges faster and reaches a lower loss value.

D.4. Loss Curve Comparison with SIREN

We plot the training loss curves of our method and SIREN on the image fitting task, as shown in Fig. 18. On the one hand, our method converges faster. On the other hand, even beyond the predefined training iterations (6,000 in our experiments), our method continues to achieve lower loss values. These results demonstrate the superior optimization efficiency of our framework.

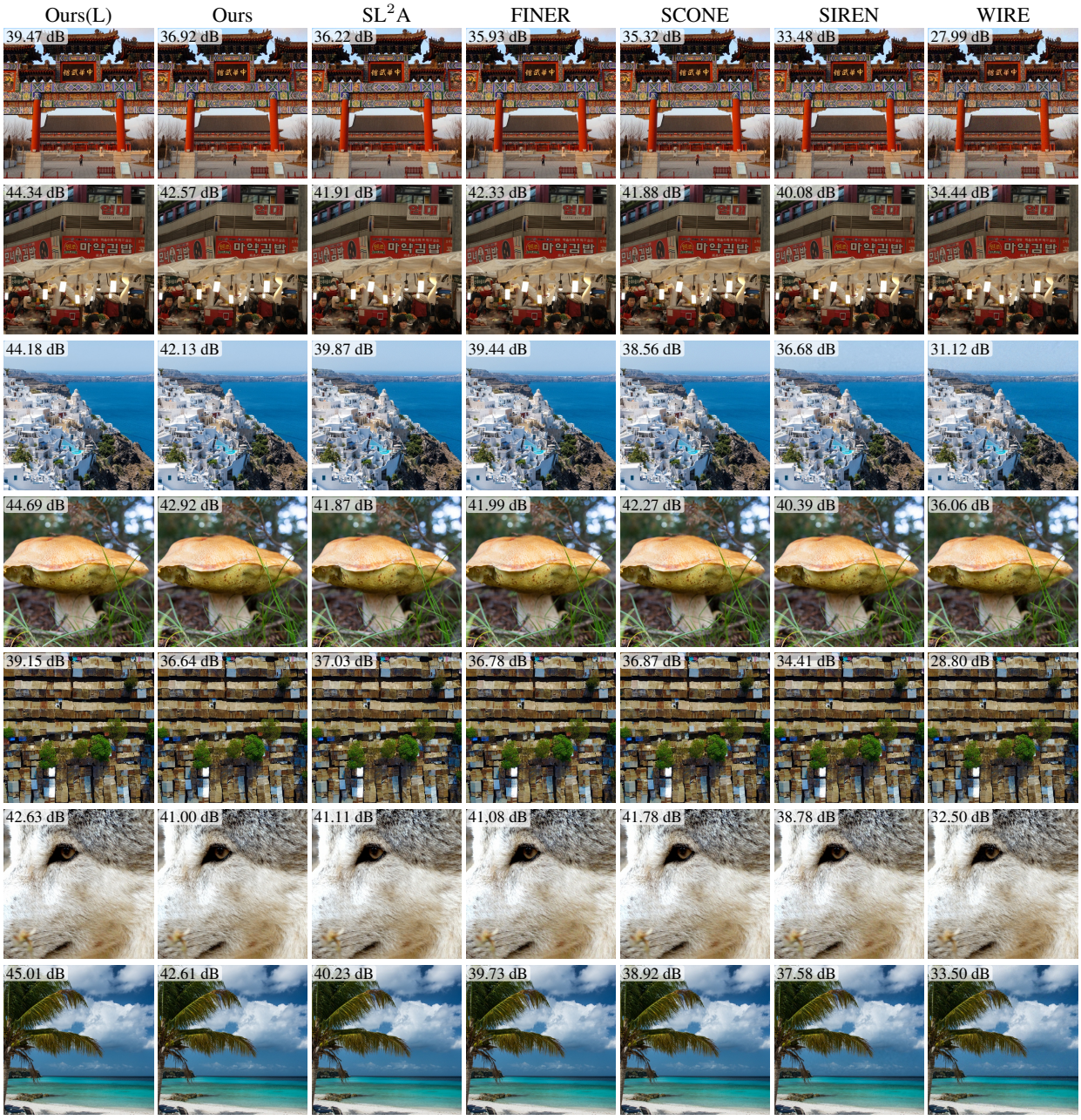


Figure 19. Visualization of the 2D image fitting task. PSNR values are shown at the upper-left corner of each image.

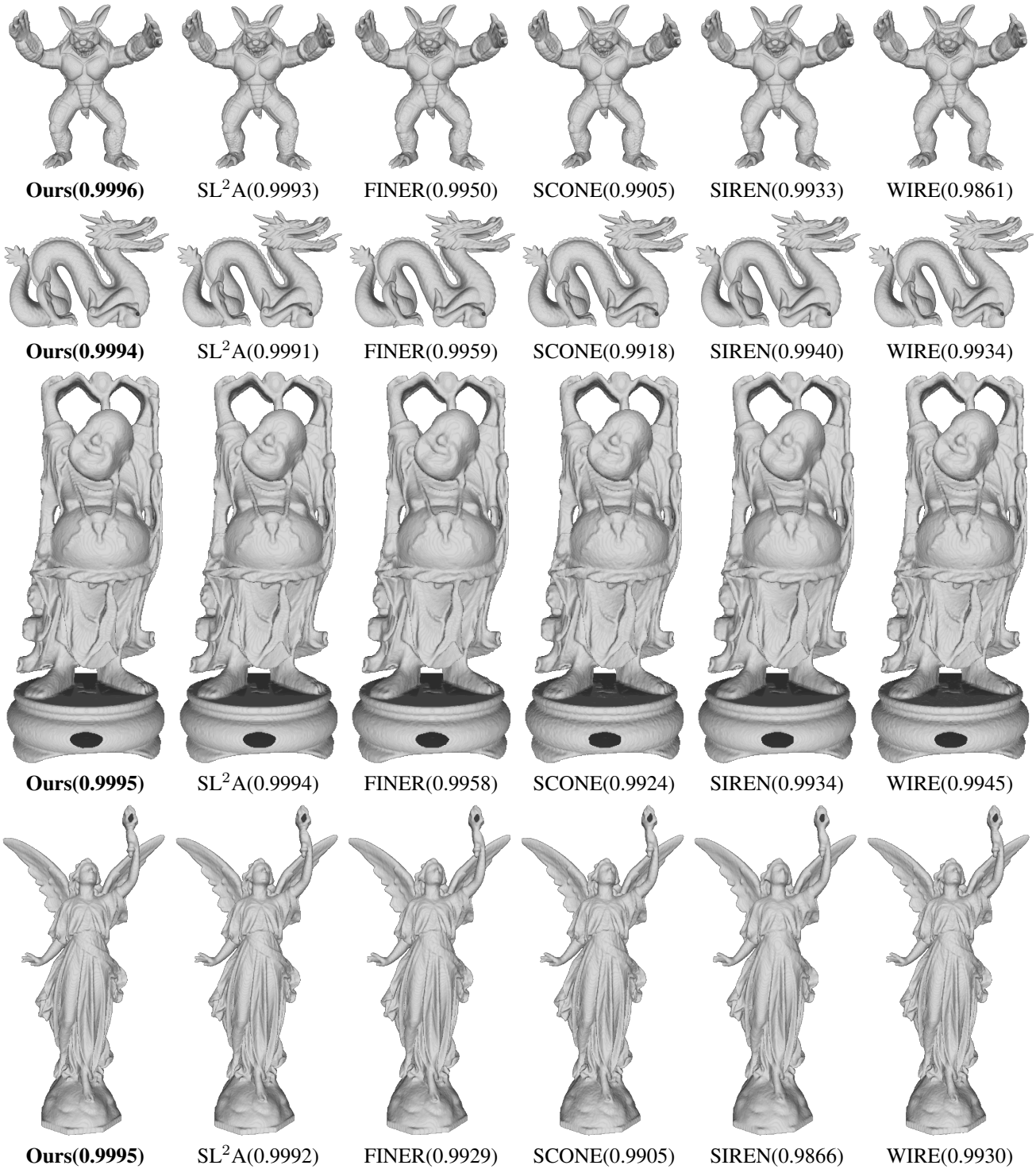


Figure 20. Visualization of the 3D shape representation task.

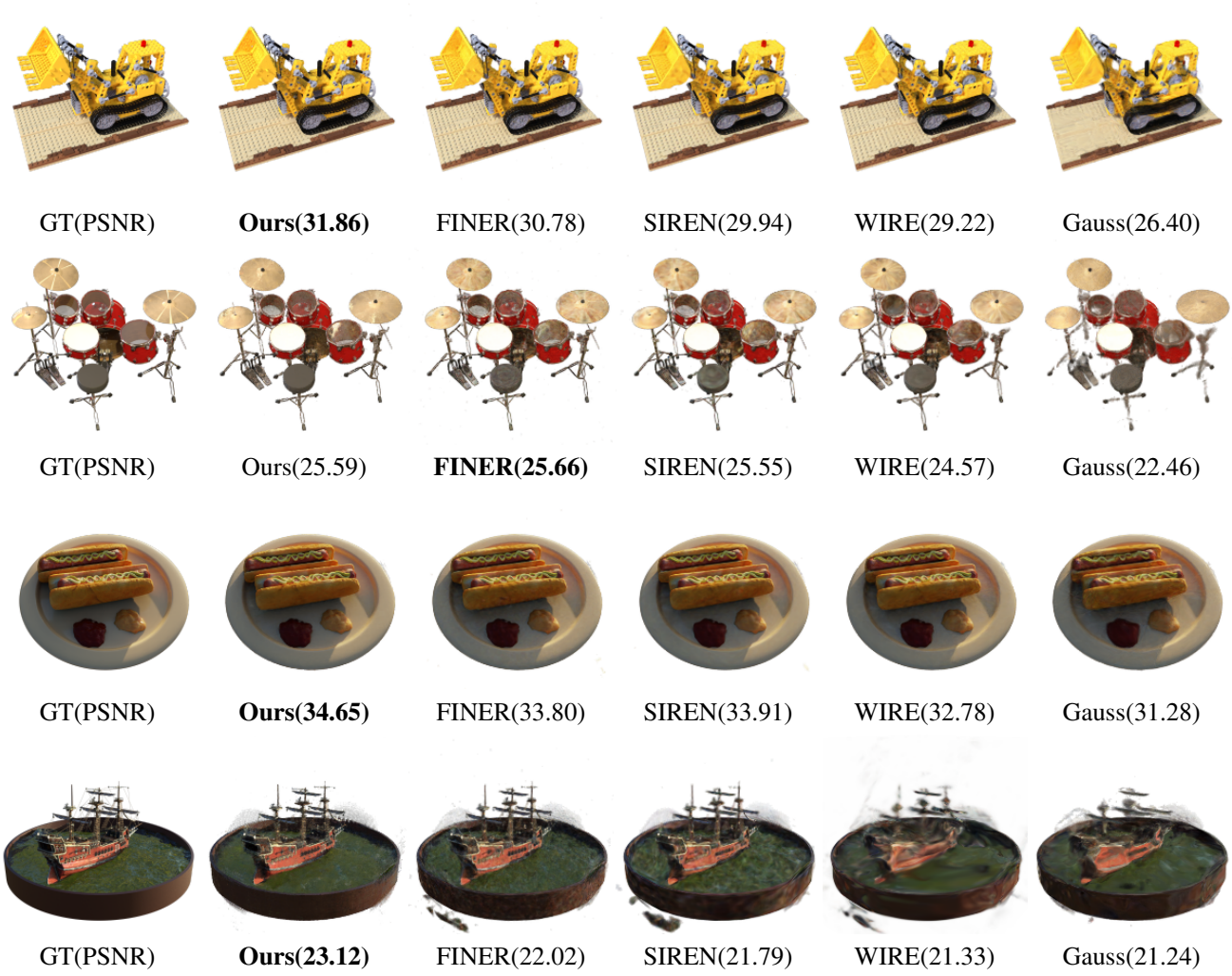


Figure 21. Visualization of the neural radiance fields task.

Ours (35.79) SL^{2}A (34.00) FINER (32.66)

SCONE (34.09) SIREN (32.42) WIRE (29.12)

Ours (30.79) SL^{2}A (26.82) FINER (26.77)

SCONE (27.68) SIREN (27.20) WIRE (23.99)

Figure 22. Visualization of the full-resolution representation results on the DIV2K images 0878 and 0891.