Regularize implicit neural representation by itself
Supplemental Material

Zhemin Li[1] Hongxia Wang*[1] Deyu Meng[2,3]
[1] National University of Defense Technology [2] Xi'an Jiaotong University
[3] Macau University of Science and Technology
lizhemin@nudt.edu.cn,wanghongxia@nudt.edu.cn,dymeng@mail.xjtu.edu.cn

A. Theoretical analysis

A.1. Fourier feature map induce a shift-invariant kernel regression

Notice that $\phi_{\text{NTK}}(\mathbf{x})$ should be shift-invariant, i.e., if we shift the training data $\{(\mathbf{x}_i, z_i)\}_{i=1}^N$ to $\{(\mathbf{x}_i + \Delta\mathbf{x}, z_i)\}_{i=1}^N$, and the corresponding kernel regression is $\phi^+_{\text{NTK}}(\mathbf{x})$, we look forward $\phi^+_{\text{NTK}}(\mathbf{x}+\Delta\mathbf{x}) = \phi_{\text{NTK}}(\mathbf{x})$. Researchers encode the shift-invariant property by a Fourier feature map $\gamma(\mathbf{x}) = \frac{1}{\sqrt{D}}[\cos \mathbf{B}\mathbf{x}^\top, \sin \mathbf{B}\mathbf{x}^\top]^\top : \mathbb{R}^d \mapsto \mathbb{R}^{2D}$ as input, where $\mathbf{x} \in \mathbb{R}^d$, $\mathbf{B} \in \mathbb{R}^{D\times d}$, and $\mathbf{B}_{ij} \sim \mathcal{N}(0, \delta)$ [3]. The NTK can be written as $h_{\text{NTK}}(\mathbf{x}_i^\top \mathbf{x}_j)$, $h_{\text{NTK}} : \mathbb{R} \mapsto \mathbb{R}$ when $\mathbf{x}_i$ on a hypersphere, so NTK with feature map can be composed as $h_{\text{NTK}}\left(\gamma(\mathbf{x}_i)^\top \gamma(\mathbf{x}_j)\right) = h_{\text{NTK}}\left(\frac{1}{D}\mathbf{1}_D^\top \cos\left(\mathbf{B}(\mathbf{x}_i - \mathbf{x}_j)\right)\right)$ which is shift-invariant. Then $\phi_{\text{NTK}}(\gamma(\mathbf{x} + \Delta\mathbf{x})) = \phi_{\text{NTK}}(\gamma(\mathbf{x}))$, and

$$\phi'_{\text{NTK}}(\mathbf{x}) = \phi_{\text{NTK}}(\gamma(\mathbf{x})) = \sum_{i=1}^N (\mathbf{H}^{-1}\mathbf{z})_i h_{\text{NTK}}(\mathbf{x}_i, \mathbf{x}),$$

where $\mathbf{H}$ is an $n \times n$ PSD matrix with entries $\mathbf{H}_{ij} = h_{\text{NTK}}(\mathbf{x}_i, \mathbf{x}_j)$.

A.2. Proof of main theorem

Proof 1 (Proof of Theorem 1)

$$\begin{aligned} k_D(\mathbf{x}_i, \mathbf{x}_j) &= h_{NTK}\left(\frac{1}{D}\mathbf{1}_D^\top \cos\left(\mathbf{B}(\mathbf{x}_i - \mathbf{x}_j)\right)\right) \\ &= h_{NTK}\left(\sum_{l=1}^D \frac{1}{D}\cos\left(\mathbf{B}_{l,:}(\mathbf{x}_i - \mathbf{x}_j)\right)\right), \end{aligned}$$

where $\boldsymbol{B}_{l,:}$ is the l-th row of $\boldsymbol{B}$.

Therefore,

$$\lim_{D\to\infty} k_D(\mathbf{x}_i, \mathbf{x}_j) = h_{NTK}\left(\mathbb{E}_{\mathbf{b}_l \sim \mathcal{N}(0,\delta)} \cos\left(\mathbf{b}^\top(\mathbf{x}_i - \mathbf{x}_j)\right)\right),$$

where $\mathbf{b} \in \mathbb{R}^d$. Furthermore, $\lim_{D\to\infty} k_D(\mathbf{x}_i, \mathbf{x}_j) = h_{NTK}(e^{-\delta^2\|\mathbf{x}_i - \mathbf{x}_j\|_2^2})$. ■

Proof 2 (Proof of Corollary 1) *As*

$$\lim_{\substack{\delta\to\infty \\ D\to\infty}} k_D(\mathbf{x}, \mathbf{X}) = \begin{cases} h(1)\mathbf{e}_i^\top & \mathbf{x} \in \{\mathbf{x}_i\}_{i=1}^N, \\ h(0)\mathbf{1}_N^\top & \mathbf{x} \notin \{\mathbf{x}_i\}_{i=1}^N, \end{cases}$$

then,

$$\lim_{\substack{\delta\to\infty \\ D\to\infty}} k_D(\mathbf{X}, \mathbf{X}) = h(0)\mathbf{1}_N\mathbf{1}_N^\top + (h(1) - h(0))\,\boldsymbol{I}_N.$$

That is, the singular value of $\lim_{\substack{\delta\to\infty \\ D\to\infty}} k_D(\mathbf{X}, \mathbf{X})$ are $h(1), h(1) - h(0), \ldots, h(1) - h(0)$. It's evidence that $k_\infty(\mathbf{X}, \mathbf{X}) = \lim_{\substack{\delta\to\infty \\ D\to\infty}} k_D(\mathbf{X}, \mathbf{X})$ is invertible when $h(1) \neq h(0)$ and $h(1) \neq 0$. Then

$$\Phi'_{NTK}(\mathbf{X}) = k_\infty(\mathbf{X}, \mathbf{X})k_\infty(\mathbf{X}, \mathbf{X})^{-1}\mathbf{z} = \mathbf{z},$$

that is $\phi'_{NTK}(\mathbf{x}_l) = z_l$.

As for $\mathbf{x} \notin \{\mathbf{x}_i\}_{i=1}^N$,

$$\begin{aligned} \Phi'_{NTK}(\mathbf{x}) &= k_\infty(\mathbf{x}, \mathbf{X}) \cdot k_\infty^{-1}(\mathbf{X}, \mathbf{X})\mathbf{z} \\ &= h(0)\mathbf{1}_N^\top k_\infty^{-1}(\mathbf{X}, \mathbf{X}) \cdot \mathbf{z}. \end{aligned} \tag{1}$$

Note that $k_\infty(\mathbf{X}, \mathbf{X})$ is a particular matrix which has same column summation that is

$$\mathbf{1}_N^\top \cdot k_\infty(\mathbf{X}, \mathbf{X}) = ((N-1)h(0) + h(1))\,\mathbf{1}_N^\top,$$

therefore its corresponding eigenvalue is $(N-1)h(0)+h(1)$. Furthermore, as

$$\frac{1}{(N-1)h(0) + h(1)}\mathbf{1}_N^\top = \mathbf{1}_N^\top \cdot k_\infty^{-1}(\mathbf{X}, \mathbf{X}),$$

we have, $\mathbf{1}_N^\top$ is the left eigenvector of $k_\infty(\mathbf{X}, \mathbf{X})^{-1}$ and the corresponding eigenvalue is $\frac{1}{(N-1)h(0)+h(1)}$. Then bring it back to Eq.1, we have

$$\Phi'_{NTK}(\mathbf{x}) = \frac{h(0)}{(N-1)h(0) + h(1)}\mathbf{1}_N^\top\mathbf{z}.$$

■

*This work was supported by the National Key Research ,Development Program (2020YFA0713504), the National Natural Science Foundation of China (61977065) and the Macao Science and Technology Development Fund (061/2020/A2).

B. Explain the proposed method step-by-step

For simplicity, we focus on a gray-scale image inpainting task to illustrate the workflow of our method.

Task: Given a partially observed image $\mathbf{X}$ on $\mathcal{X}$, where $\mathcal{X} \subseteq \mathcal{G} = \left\{(\frac{i}{m}, \frac{j}{n}) | i \in \{1, 2, \cdots, m\}, j \in \{1, 2, \cdots, n\}\right\}$, i.e., $\mathcal{Z} = \left\{\mathbf{X}_{i,j} \mid (\frac{i}{m}, \frac{j}{n}) \in \mathcal{X}\right\}$, find $\mathbf{X}$ on unobserved $\mathcal{G}\backslash\mathcal{X}$.

Input: Training set $\mathcal{X} \times \mathcal{Z}$; initial network parameters $\boldsymbol{\theta}_a(0) = \{\boldsymbol{\theta}(0), \boldsymbol{\theta}_r(0), \boldsymbol{\theta}_c(0)\}$; super-parameters λ_r, λ_c; iteration step $t = 0$.

Step 1: *Calculate loss function.* Loss function is $\mathcal{L}_a(\boldsymbol{\theta}_a) = \mathcal{L}(\boldsymbol{\theta}(t), \mathcal{X}, \mathcal{Z}) + \lambda_r \mathcal{R}(\boldsymbol{\theta}_r(t)) + \lambda_c \mathcal{R}(\boldsymbol{\theta}_c(t))$, where $\mathcal{L} = \sum_{(\mathbf{x}_i, z_i) \in \mathcal{X} \times \mathcal{Z}} \left\|\phi_{\boldsymbol{\theta}(t)}(\mathbf{x}_i) - z_i\right\|_2^2$, $\mathcal{R}(\boldsymbol{\theta}_r(t)) = \operatorname{tr}\left(\mathbf{X}^\top \mathbf{L}(\boldsymbol{\theta}_r(t))\mathbf{X}\right)$ and $\mathcal{R}(\boldsymbol{\theta}_c(t)) = \operatorname{tr}\left(\mathbf{X}\mathbf{L}(\boldsymbol{\theta}_c(t))\mathbf{X}^\top\right)$ measure the similarity between rows and columns in image respectively.

Step 2: *Update parameters.* Minimize $\mathcal{L}_a$ by updating parameters $\boldsymbol{\theta}_a(t)$ with optimization algorithm such as Adam. Iteration stops at $t = T$ when $\mathcal{L}_a(\boldsymbol{\theta}_a(T))$ is smaller than some given precision.

Step 3: *Output estimation.* The pixel value of $\mathbf{X}_{ij}$ on $\mathcal{G}\backslash\mathcal{X}$ is predicted by $\phi_{\boldsymbol{\theta}(T)}(\frac{i}{m}, \frac{j}{n})$.

For high-dimensional data such as video, the regularizer captures the similarity between the vectorized frames.

C. Additional experiments

As a general image representation model, our method can be readily applied to other image tasks, including those higher dimensional ones. Table 1 shows that INRR outperforms INR in image denoising under variant noise types. Besides, we have also verified that INRR shows its power in video frame interpolation and RGB image inpainting.

Table 1. PSNR (dB) of denoised images by INR and INRR on (a) Baboon, (b) Man, (c) Barbara, (d) Boats and (e) Cameraman.

Noise Type	Method	(a)	(b)	(c)	(d)	(e)
Gaussian ($\sigma = 10$)	INR	28.5	29.3	30.5	30.8	33.1
	INRR	**29.0**	**29.7**	**30.6**	**31.2**	**33.3**
Salt & Peper ($r = 0.95$)	INR	21.9	23.0	25.4	24.7	25.9
	INRR	**22.0**	**23.1**	**25.8**	**24.9**	**25.9**
Poisson ($\lambda = 50$)	INR	23.2	24.9	27.0	25.9	27.7
	INRR	**23.6**	**25.2**	**27.1**	**26.3**	**28.1**

Here we test INRR on video interpolation, where the data is represented by $\phi_{\boldsymbol{\theta}}(x, y, t) = [r, g, b] : \mathbb{R}^3 \mapsto \mathbb{R}^3$. Each frame is an RGB image, as shown in Figure 1 (a). The tested video is a scene of water droplets that owns 202 frames. We sample 21 frames uniformly as training data; the rest are test data. Figure 1 (b) shows that INRR can capture the non-local self-similarity between different frames. The average PSNR of INRR is 37.5 dB, while vanilla INR is 36.8 dB. Furthermore, we have validated the inpainting performance on a dataset BSD100 [2] which includes 100 RGB images. The sampling mode is the same as in Figure 2(a) in the paper. The average PSNR of INRR is 28.8 dB, while vanilla INR is 27.1 dB.

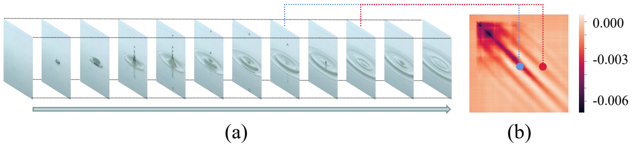

Figure 1. The similarity between different frames (a) can be captured by $\mathbf{L}$ (L392) in INRR (b) which is helpful to frame interpolation.

D. Smoothness of Laplacian matrix

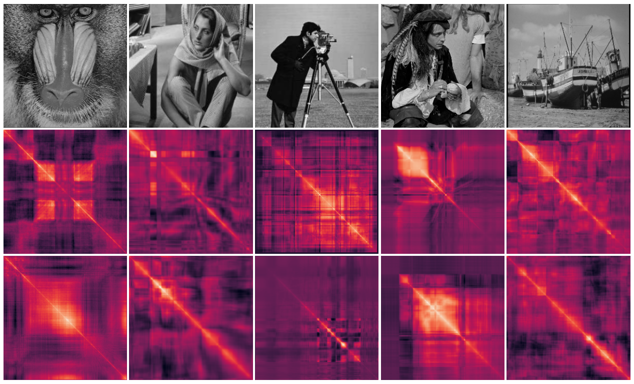

Figure 2. The first row show five different 256 × 256 gray-scale images. The second row shows the column covariance matrix, and the third row shows the row covariance matrix.

We first calculate the covariance matrix of the columns and the rows in Figure 2. The covariance matrix of $\mathbf{X}$ is $\mathbf{C}(\mathbf{X})$, where $\mathbf{C}_{ij} = \mathbb{E}[\mathbf{X}_{:,i} - \mathbb{E}(\mathbf{X}_{:,i})][\mathbf{X}_{:,j} - \mathbb{E}(\mathbf{X}_{:,j})]$, which measures the similarity among the columns. While the similarity among rows is $\mathbf{C}(\mathbf{X}^\top)$. As we can see, all the covariance matrices of various images are locally smoothly. Dong et al. proposed to utilize the smoothness of the Laplacian matrix by an extra regularizer [1]. We use an INR to encode the smoothness in such a Laplacian matrix implicitly; that is, our proposed INRR combines the self-similarity and smoothness of the Laplacian matrix at the same time.

Because $g(\boldsymbol{\theta}; \mathbf{u})$ is an INR which is a smooth FCN about $\mathbf{u}$, $\mathbf{A}(\boldsymbol{\theta}) = \frac{\exp\left(g^\top(\boldsymbol{\theta};\mathbf{u})g(\boldsymbol{\theta};\mathbf{u})\right)}{\mathbf{1}^\top \exp(g^\top(\boldsymbol{\theta};\mathbf{u})g(\boldsymbol{\theta};\mathbf{u}))\mathbf{1}}$ is smooth according to its expression. It means that a slight change of $\mathbf{u}$ generally leads to a slight change of $\mathbf{L}(\boldsymbol{\theta}) = \mathbf{D}(\boldsymbol{\theta}) - \mathbf{A}(\boldsymbol{\theta})$, which can be controlled by a Lipschitz constant. So we conclude that $\mathbf{A}(\boldsymbol{\theta})$ smoothes $\mathbf{L}(\boldsymbol{\theta})$, which is different from the vanilla $\mathbf{L}$. Furthermore, we can deduce a conclusion similar to Theorem 1 that the smoothness of $\mathbf{L}(\boldsymbol{\theta})$ is controlled by the δ of $g(\boldsymbol{\theta}; \cdot)$. Smaller δ leads to a smoother result. Moreover, INRR degenerates to AIR when $\delta \to \infty$.

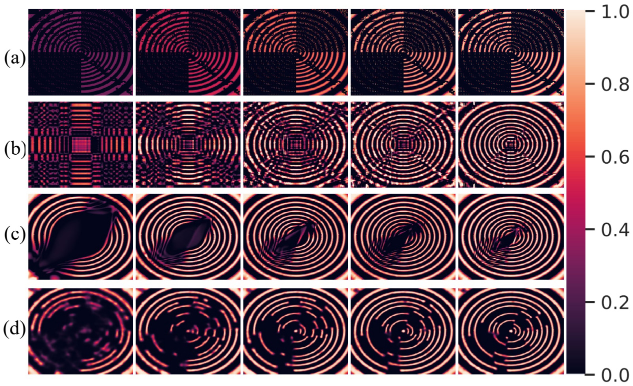

Figure 3. Fitting a 256×256 synthetic data with a combine missing at different training step with (a) single matrix (DMF with only one factor), (b) DMF with three factors, (c) ReLU neural network, and (d) SIREN. The effective rank of the fitted matrix is shown in (e).

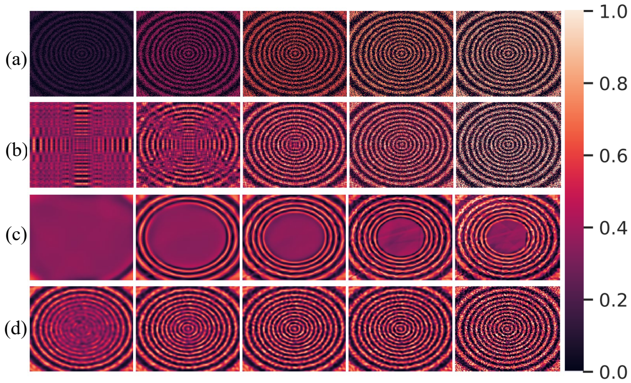

Figure 4. Fitting a 256×256 synthetic data with Gaussian noise at different training step with (a) single matrix (DMF with only one factor), (b) DMF with three factors, (c) ReLU neural network, and (d) SIREN. The effective rank of the fitted matrix is shown in (e).

E. Implicit bias

The neural network tends to converge to a good solution and may suffer from over-fitting with the training goes. Researchers explain this phenomenon by the implicit bias of neural networks. We show the implicit bias by fitting the synthetic data, which is sampled from function $s(x, y) = \sin\left(25\pi \sin\left(\frac{\pi}{3} \cdot \sqrt{x^2 + y^2}\right)\right)$, where $\{(x_i, y_j)|i, j\}$ is a uniform 256×256 grid on $[-1, 1] \times [-1, 1]$, where the local frequency of the synthetic data increases from boundary to center. We show two tasks on such synthetic data: fitting the incomplete data and the noisy data in Figure 3 and Figure 4, respectively.

As Figure 3(a) shows, the single layer DMF fits all pixels without bias, and the pixels of the fitted image increase gradually. While Figure 3(b) shows the low-rank bias of the three-layer DMF fitting the synthetic data from low-rank to high-rank. Figure 3(c,d) shows that a fully connected neural network's bias is related to the data frequency and sampling rate. With the bias mentioned above, it is possible to complete an image without an extra explicit regularizer. Similarly, these neural network has similar phenomenon when fitting the noisy data.

F. Recovered image

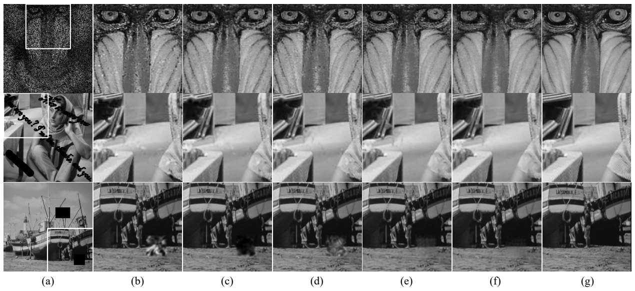

Figure 5. Result of image inpainting with three types of missing data by different regularized INR including (b) INR without regularization, (c) with TV, (d) L_2, (e) AIR, (f) INRR, and (g) original image. The hyper-parameters of benchmark models and algorithms are adopted from the original paper.

We show the recovered image in Figure 5.

References

[1] Bin Dong, Haocheng Ju, Yiping Lu, and Zuoqiang Shi. Cure: Curvature regularization for missing data recovery. *Siam Journal on Imaging Sciences*, 2019. 2

[2] David Martin, Charless C. Fowlkes, D. Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. *International Conference on Computer Vision*, 2001. 2

[3] 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. *NeurIPS*, 2020. 1