Supplementary Material:
A Structured Dictionary Perspective on Implicit Neural Representations

Contents

S1. Deferred proofs

S1.1. Proof of Theorem 1

We provide here the proof of Theorem 1 which gives an explicit expression to the expressive power of INRs. However, before we delve deeper in this proof we will prove a few useful lemmas.

S1.1.1 Preliminary lemmas

Lemma 1. *Let $\{\boldsymbol{\omega}_k^{(1)} \in \mathbb{R}^D\}_{k\in\mathcal{K}}$ and $\{\boldsymbol{\omega}_j^{(2)} \in \mathbb{R}^D\}_{j\in\mathcal{J}}$, and $\{\phi_k^{(1)} \in \mathbb{R}\}_{k\in\mathcal{K}}$ and $\{\phi_j^{(2)} \in \mathbb{R}\}_{j\in\mathcal{J}}$ be two collections of frequency vectors and scalar phases, respectively, indexed by the sets $\mathcal{K},\mathcal{J} \subseteq \mathbb{N}$. Furthermore, let $\{\beta_k^{(1)} \in \mathbb{R}\}_{k\in\mathcal{K}}$ and $\{\beta_j^{(2)} \in \mathbb{R}\}_{j\in\mathcal{J}}$ be two sets of scalar coefficients and $\boldsymbol{r} \in \mathbb{R}^D$. Then,*

$$\left(\sum_{k\in\mathcal{K}} \beta_k^{(1)} \cos\left(\left\langle \boldsymbol{\omega}_k^{(1)}, \boldsymbol{r}\right\rangle + \phi_k^{(1)}\right)\right)\left(\sum_{j\in\mathcal{J}} \beta_j^{(2)} \cos\left(\left\langle \boldsymbol{\omega}_j^{(2)}, \boldsymbol{r}\right\rangle + \phi_j^{(2)}\right)\right) = \sum_{\boldsymbol{\omega}'\in\mathcal{D}} \tilde{\beta}_{\boldsymbol{\omega}'} \cos(\langle \boldsymbol{\omega}', \boldsymbol{r}\rangle + \tilde{\phi}_{\boldsymbol{\omega}'}) \tag{1}$$

where

$$\mathcal{D}\left(\left\{\boldsymbol{\omega}_k^{(1)}\right\}_{k\in\mathcal{K}}, \left\{\boldsymbol{\omega}_j^{(2)}\right\}_{j\in\mathcal{J}}\right) = \left\{\boldsymbol{\omega}' = \boldsymbol{\omega}_k^{(1)} \pm \boldsymbol{\omega}_j^{(2)} \middle| k \in \mathcal{K}, j \in \mathcal{J}\right\} \tag{2}$$

for some $\left\{\tilde{\phi}_{\boldsymbol{\omega}'} \in \mathbb{R} \middle| \boldsymbol{\omega}' \in \mathcal{D}\right\}$, $\left\{\tilde{\beta}_{\boldsymbol{\omega}'} \in \mathbb{R} \middle| \boldsymbol{\omega}' \in \mathcal{D}\right\}$.

Proof.

$$\begin{aligned}
&\left(\sum_{k\in\mathcal{K}} \beta_k^{(1)} \cos\left(\left\langle \boldsymbol{\omega}_k^{(1)}, \boldsymbol{r}\right\rangle + \phi_k^{(1)}\right)\right)\left(\sum_{j\in\mathcal{J}} \beta_j^{(2)} \cos\left(\left\langle \boldsymbol{\omega}_j^{(2)}, \boldsymbol{r}\right\rangle + \phi_j^{(2)}\right)\right) \\
=&\sum_{k\in\mathcal{K}}\sum_{j\in\mathcal{J}} \beta_k^{(1)}\beta_j^{(2)} \cos\left(\left\langle \boldsymbol{\omega}_k^{(1)}, \boldsymbol{r}\right\rangle + \phi_k^{(1)}\right)\cos\left(\left\langle \boldsymbol{\omega}_j^{(2)}, \boldsymbol{r}\right\rangle + \phi_j^{(2)}\right) \\
=&\sum_{k\in\mathcal{K}}\sum_{j\in\mathcal{J}} \beta_k^{(1)}\beta_j^{(2)} \frac{1}{2}\left(\cos\left(\left\langle \boldsymbol{\omega}_k^{(1)}, \boldsymbol{r}\right\rangle + \left\langle \boldsymbol{\omega}_j^{(2)}, \boldsymbol{r}\right\rangle + \phi_k^{(1)} + \phi_j^{(2)}\right) + \cos\left(\left\langle \boldsymbol{\omega}_k^{(1)}, \boldsymbol{r}\right\rangle - \left\langle \boldsymbol{\omega}_j^{(2)}, \boldsymbol{r}\right\rangle + \phi_k^{(1)} - \phi_j^{(2)}\right)\right) \\
=&\sum_{k\in\mathcal{K}}\sum_{j\in\mathcal{J}} \beta_k^{(1)}\beta_j^{(2)} \frac{1}{2}\left(\cos\left(\left\langle \boldsymbol{\omega}_k^{(1)} + \boldsymbol{\omega}_j^{(2)}, \boldsymbol{r}\right\rangle + \phi_k^{(1)} + \phi_j^{(2)}\right) + \cos\left(\left\langle \boldsymbol{\omega}_k^{(1)} - \boldsymbol{\omega}_j^{(2)}, \boldsymbol{r}\right\rangle + \phi_k^{(1)} - \phi_j^{(2)}\right)\right) \\
=&\sum_{\boldsymbol{\omega}'\in\mathcal{D}} \tilde{\beta}_{\boldsymbol{\omega}'} \cos(\langle \boldsymbol{\omega}', \boldsymbol{r}\rangle + \tilde{\phi}_{\boldsymbol{\omega}'})
\end{aligned} \tag{3}$$

□

Lemma 2. *Let $\{\boldsymbol{\omega}_j \in \mathbb{R}^D\}_{j\in\mathcal{J}}$ and $\{\phi_j \in \mathbb{R}\}_{j\in\mathcal{J}}$ be a collection of frequency vectors and scalar phases, respectively, indexed by the set $\mathcal{J} \subseteq \mathbb{N}$. Furthermore, $\{\beta_j \in \mathbb{R}\}_{j\in\mathcal{J}}$ be a set of scalar coefficients, and let $k \in \mathbb{N}$ Then,*

$$\left(\sum_{j\in\mathcal{J}} \beta_j \cos\left(\langle \boldsymbol{\omega}_j, \boldsymbol{r}\rangle + \phi_j\right)\right)^k = \sum_{\boldsymbol{\omega}'\in\mathcal{H}_k} \tilde{\beta}_{\boldsymbol{\omega}'} \cos(\langle \boldsymbol{\omega}', \boldsymbol{r}\rangle + \tilde{\phi}_{\boldsymbol{\omega}'}) \tag{4}$$

where

$$\mathcal{H}_k\left(\{\boldsymbol{\omega}_j\}_{j\in\mathcal{J}}\right) \subseteq \tilde{\mathcal{H}}_k\left(\{\boldsymbol{\omega}_j\}_{j\in\mathcal{J}}\right) := \left\{\boldsymbol{\omega}' = \sum_{j\in\mathcal{J}} c_j\boldsymbol{\omega}_j \middle| c_j \in \mathbb{Z} \wedge \sum_{j\in\mathcal{J}} |c_j| \leq k\right\} \tag{5}$$

Note that we will often use the notation $\mathcal{H}_k$ and $\tilde{\mathcal{H}}_k$ instead of explicitly writing the dependence on the set $\left(\{\boldsymbol{\omega}_j\}_{j\in\mathcal{J}}\right)$ when it is clear from the context.

Proof. The statement trivially holds for $k = 1$. Assume it also holds for k, then

$$\begin{aligned}\left(\sum_{j\in\mathcal{J}} \beta_j \cos\left(\langle \boldsymbol{\omega}_j, \boldsymbol{r}\rangle + \phi_j\right)\right)^{k+1} &= \left(\sum_{j\in\mathcal{J}} \beta_j \cos\left(\langle \boldsymbol{\omega}_j, \boldsymbol{r}\rangle + \phi_j\right)\right)^{k} \left(\sum_{j\in\mathcal{J}} \beta_j \cos\left(\langle \boldsymbol{\omega}_j, \boldsymbol{r}\rangle + \phi_j\right)\right) &(6)\\ &= \left(\sum_{\boldsymbol{\omega}'\in\mathcal{H}_k} \tilde{\beta}_{\boldsymbol{\omega}'} \cos(\langle \boldsymbol{\omega}', \boldsymbol{r}\rangle + \tilde{\phi}_{\boldsymbol{\omega}'})\right) \left(\sum_{j\in\mathcal{J}} \beta_j \cos\left(\langle \boldsymbol{\omega}_j, \boldsymbol{r}\rangle + \phi_j\right)\right) &(7)\\ &= \sum_{\boldsymbol{\omega}'\in\mathcal{D}\left\{\mathcal{H}_k, \{\boldsymbol{\omega}_j\}_{j\in\mathcal{J}}\right\}} \beta'_{\boldsymbol{\omega}'} \cos(\langle \boldsymbol{\omega}', \boldsymbol{r}\rangle + \phi'_{\boldsymbol{\omega}'}) &(8)\\ &= \sum_{\boldsymbol{\omega}'\in\mathcal{H}_{k+1}} \beta'_{\boldsymbol{\omega}'} \cos(\langle \boldsymbol{\omega}', \boldsymbol{r}\rangle + \phi'_{\boldsymbol{\omega}'}) &(9)\\ & &(10)\end{aligned}$$

where Eq. (7) holds by assumption and Eq. (8) holds because of the previous lemma. Moreover we have

$$\begin{aligned}\mathcal{H}_{k+1} = D\left\{\mathcal{H}_k, \{\boldsymbol{\omega}_i\}_{i\in\mathcal{J}}\right\} &= \left\{\boldsymbol{\omega}' = \boldsymbol{\omega}_h \pm \boldsymbol{\omega}_i \middle| \boldsymbol{\omega}_h \in \mathcal{H}_k, i \in \mathcal{J}\right\} &(11)\\ &\subseteq \left\{\boldsymbol{\omega}' = \sum_{j\in\mathcal{J}} c_j\boldsymbol{\omega}_j \pm \boldsymbol{\omega}_i \middle| c_j \in \mathbb{Z} \wedge \sum_{j\in\mathcal{J}} |c_j| \leq k, i \in \mathcal{J}\right\} &(12)\\ &\subseteq \left\{\boldsymbol{\omega}' = \sum_{j\in\mathcal{J}} c_j\boldsymbol{\omega}_j \middle| c_j \in \mathbb{Z} \wedge \sum_{j\in\mathcal{J}} |c_j| \leq k+1\right\} &(13)\\ & &(14)\end{aligned}$$

So Eq. (4) holds for $k + 1$ as well. Then by induction Eq. (4) holds $\forall k \in \mathbb{N}$ □

S1.1.2 Main proof

Recall that we are interested in understanding the expressive power of INR architectures that can be decomposed into a mapping function $\gamma : \mathbb{R}^D \to \mathbb{R}^T$ followed by a multilayer perceptron (MLP), with weights $\boldsymbol{W}^{(\ell)} \in \mathbb{R}^{F_{\ell-1}\times F_\ell}$, bias $\boldsymbol{b}^{(\ell)} \in \mathbb{R}^{F_\ell}$, and activation function $\rho^{(\ell)} : \mathbb{R} \to \mathbb{R}$, applied elementwise; at each layer $\ell = 1, \dots, L-1$. That is, if we denote by $\boldsymbol{z}^{(\ell)}$ each layers post activation, most INR architectures compute

$$\begin{aligned}\boldsymbol{z}^{(0)} &= \gamma(\boldsymbol{r}),\\ \boldsymbol{z}^{(\ell)} &= \rho^{(\ell)}\left(\boldsymbol{W}^{(\ell)}\boldsymbol{z}^{(\ell-1)} + \boldsymbol{b}^{(\ell)}\right),\ \ell = 1, \dots, L-1 \qquad (15)\\ f_{\boldsymbol{\theta}}(\boldsymbol{r}) &= \boldsymbol{W}^{(L)}\boldsymbol{z}^{(L-1)} + \boldsymbol{b}^{(L)}.\end{aligned}$$

Based on this architecture we can prove the following theorem.

Theorem 1. *Let $f_{\boldsymbol{\theta}} : \mathbb{R}^D \to \mathbb{R}$ be an INR of the form of Eq.* (15) *with $\rho^{(\ell)}(z) = \sum_{k=0}^{K} \alpha_k z^k$ for $\ell > 1$. Furthermore, let $\boldsymbol{\Omega} = [\boldsymbol{\Omega}_0, \dots, \boldsymbol{\Omega}_{T-1}]^\top \in \mathbb{R}^{T\times D}$ and $\boldsymbol{\phi} \in \mathbb{R}^T$ denote the matrix of frequencies and vector of phases, respectively, used to map the input coordinate $\boldsymbol{r} \in \mathbb{R}^D$ to $\gamma(\boldsymbol{r}) = \sin(\boldsymbol{\Omega}\boldsymbol{r} + \boldsymbol{\phi})$. This architecture can only represent functions of the form*

$$f_{\boldsymbol{\theta}}(r) = \sum_{\boldsymbol{\omega}'\in\mathcal{H}(\boldsymbol{\Omega})} c_{\boldsymbol{\omega}'} \sin\left(\langle \boldsymbol{\omega}', \boldsymbol{r}\rangle + \phi_{\boldsymbol{\omega}'}\right), \qquad (16)$$

where

$$\mathcal{H}(\boldsymbol{\Omega}) \subseteq \left\{\boldsymbol{\omega}' = \sum_{t=0}^{T-1} s_t\boldsymbol{\Omega}_t \;\middle|\; s_t \in \mathbb{Z} \wedge \sum_{t=0}^{T-1} |s_t| \leq K^{L-1}\right\}. \qquad (17)$$

Proof. We will prove the statement by induction. To that end, let us denote the preactivation vector at each layer as $\boldsymbol{v}^{(\ell)}$, i.e. $\boldsymbol{z}^{(\ell)} = \rho^{(\ell)}\left(\boldsymbol{v}^{(\ell)}\right)$. We will first derive the expressions for the base case.

Base case Consider the preactivation of a node at the first layer of the neural network for any mapping of the form in Eq. (15). Then

$$\boldsymbol{v}_j^{(1)} = \boldsymbol{W}_j^{(1)}\gamma(\boldsymbol{r}) = \sum_{t=0}^{T-1} b_{tj}\cos\left(\langle\boldsymbol{\Omega}_t, \boldsymbol{r}\rangle + \phi_{tj}\right) \tag{18}$$

with some $b_{tj} \in \mathbb{R}$ and $\phi_{tj} \in \mathbb{R}$ depending on the first layer weights connected to that node. Also note that interchanging sines with cosines only affects the phase terms.

Therefore, using the result of Lemma 2, and after applying the activation function, the output of each node at the first layer is given by

$$\begin{aligned}\boldsymbol{z}_j^{(1)} = \rho^{(1)}\left(\boldsymbol{v}_j^{(1)}\right) = \sum_{k=0}^{K}\alpha_k\left(\boldsymbol{v}_j^{(1)}\right)^k &= \sum_{k=0}^{K}\alpha_k\left(\sum_{t=0}^{T-1} b_{tj}\cos\left(\langle\boldsymbol{\Omega}_t, \boldsymbol{r}\rangle + \phi_{tj}\right)\right)^k & (19)\\ &= \sum_{k=0}^{K}\alpha_k\sum_{\boldsymbol{\omega}'_k\in\mathcal{H}_k}\beta_{\boldsymbol{\omega}'_k}\cos(\langle\boldsymbol{\omega}'_k, \boldsymbol{r}\rangle + \phi_{\boldsymbol{\omega}'_k}) & (20)\\ &= \sum_{\boldsymbol{\omega}_k\in\mathcal{H}'_K}\tilde{\beta}_{\boldsymbol{\omega}_k}\cos(\langle\boldsymbol{\omega}_k, \boldsymbol{r}\rangle + \tilde{\phi}_{\boldsymbol{\omega}_k}) & (21)\end{aligned}$$

where $\mathcal{H}'_K := \bigcup_{j=1}^{K}\mathcal{H}_j$ and we use the definitions of $\mathcal{H}_k$ and $\tilde{\mathcal{H}}_k$ in Lemma 2. Therefore, since $\forall k \mathcal{H}_k \subseteq \tilde{\mathcal{H}}_k$ by construction, and $\forall j \leq k\ \mathcal{H}_j \subseteq \tilde{\mathcal{H}}_k$; then it holds that $\mathcal{H}'_K \subseteq \tilde{\mathcal{H}}_K$, i.e.,

$$\mathcal{H}'_K \subseteq \tilde{\mathcal{H}}_K = \left\{\boldsymbol{\omega}' = \sum_{t=0}^{T-1}s_t\boldsymbol{\Omega}_t \;\middle|\; s_t\in\mathbb{Z}\wedge\sum_{t=0}^{T-1}|s_t| \leq K\right\}. \tag{22}$$

Induction step Assume the output of the nodes at layer ℓ satisfy the following expression:

$$\boldsymbol{z}_j^{(\ell)} = \sum_{\omega'\in\mathcal{H}^{(\ell)}(\boldsymbol{\Omega})} c_{\boldsymbol{\omega}',j}\sin\left(\langle\boldsymbol{\omega}', \boldsymbol{r}\rangle + \phi_{\boldsymbol{\omega}',j}\right) \tag{23}$$

where

$$\mathcal{H}^{(\ell)} \subseteq \tilde{\mathcal{H}}_{K^\ell} = \left\{\boldsymbol{\omega}' = \sum_{t=0}^{T-1}s_t\boldsymbol{\Omega}_t \;\middle|\; s_t\in\mathbb{Z}\wedge\sum_{t=0}^{T-1}|s_t| \leq K^\ell\right\}. \tag{24}$$

Then, the preactivation of any node at the $(\ell+1)^{th}$ layer can be expressed as:

$$v_j^{(\ell+1)} = \sum_{\boldsymbol{\omega}'\in\mathcal{H}^{(\ell)}(\boldsymbol{\Omega})} b_{\boldsymbol{\omega}',j}\sin\left(\langle\boldsymbol{\omega}', \boldsymbol{r}\rangle + \tilde{\phi}_{\boldsymbol{\omega}',j}\right) \tag{25}$$

since the sum of cosines with the same frequency only result in a cosine with the same frequency but with a modified phase and amplitude. Hence, after applying the activation function the output of the j^{th} node at the $(\ell+1)^{th}$ layer can be written as:

$$\boldsymbol{z}_j^{(\ell+1)} = \rho^{(\ell+1)}\left(\boldsymbol{v}_j^{(\ell+1)}\right) = \sum_{k=0}^{K}\alpha_k\left(\boldsymbol{v}_j^{(\ell+1)}\right)^k = \sum_{k=0}^{K}\alpha_k\left(\sum_{\boldsymbol{\omega}'\in\mathcal{H}^{(\ell)}(\boldsymbol{\Omega})} b_{\boldsymbol{\omega}',j}\sin\left(\langle\boldsymbol{\omega}', \boldsymbol{r}\rangle + \tilde{\phi}_{\boldsymbol{\omega}',j}\right)\right)^k \tag{26}$$

Let us inspect the term $\left(\sum_{\boldsymbol{\omega}'\in\mathcal{H}^{(\ell)}(\boldsymbol{\Omega})} b_{\boldsymbol{\omega}',j}\sin\left(\langle\boldsymbol{\omega}',\boldsymbol{r}\rangle+\tilde{\phi}_{\boldsymbol{\omega}',j}\right)\right)^k$. Instead of directly applying Lemma 2, we will leverage the fact that all the frequencies $\boldsymbol{\omega}'\in\mathcal{H}^{(\ell)}$ share a similar structure. More precisely, they all can be represented as a sum of the frequencies in the set $\boldsymbol{\Omega}$. To that end, let us show the following intermediate result:

$$\left(\sum_{\boldsymbol{\omega}'\in\mathcal{H}^{(\ell)}(\boldsymbol{\Omega})} b_{\boldsymbol{\omega}',j}\sin\left(\langle\boldsymbol{\omega}',\boldsymbol{r}\rangle+\tilde{\phi}_{\boldsymbol{\omega}',j}\right)\right)^k = \sum_{\boldsymbol{\omega}'\in\mathcal{H}_k^{(\ell)}(\boldsymbol{\Omega})} \tilde{b}_{\boldsymbol{\omega}',j}\sin\left(\langle\boldsymbol{\omega}',\boldsymbol{r}\rangle+\tilde{\tilde{\phi}}_{\boldsymbol{\omega}',j}\right) \tag{27}$$

where $\mathcal{H}_k^{(\ell)}\subseteq\tilde{\mathcal{H}}_{kK^\ell}$. The base case for $k=1$ holds trivially. Now assume Eq. (27) holds for k, then

$$\begin{aligned}
\left(\sum_{\boldsymbol{\omega}'\in\mathcal{H}^{(\ell)}(\boldsymbol{\Omega})} b_{\boldsymbol{\omega}',j}\sin\left(\langle\boldsymbol{\omega}',\boldsymbol{r}\rangle+\tilde{\phi}_{\boldsymbol{\omega}',j}\right)\right)^{k+1} &= \left(\sum_{\boldsymbol{\omega}'\in\mathcal{H}^{(\ell)}(\boldsymbol{\Omega})} b_{\boldsymbol{\omega}',j}\sin\left(\langle\boldsymbol{\omega}',\boldsymbol{r}\rangle+\tilde{\phi}_{\boldsymbol{\omega}',j}\right)\right)^{k}\left(\sum_{\boldsymbol{\omega}'\in\mathcal{H}^{(\ell)}(\boldsymbol{\Omega})} b_{\boldsymbol{\omega}',j}\sin\left(\langle\boldsymbol{\omega}',\boldsymbol{r}\rangle+\tilde{\phi}_{\boldsymbol{\omega}',j}\right)\right) && (28)\\
&= \left(\sum_{\boldsymbol{\omega}'\in\mathcal{H}_k^{(\ell)}(\boldsymbol{\Omega})} \tilde{b}_{\boldsymbol{\omega}',j}\sin\left(\langle\boldsymbol{\omega}',\boldsymbol{r}\rangle+\tilde{\tilde{\phi}}_{\boldsymbol{\omega}',j}\right)\right)\left(\sum_{\boldsymbol{\omega}'\in\mathcal{H}^{(\ell)}(\boldsymbol{\Omega})} b_{\boldsymbol{\omega}',j}\sin\left(\langle\boldsymbol{\omega}',\boldsymbol{r}\rangle+\tilde{\phi}_{\boldsymbol{\omega}',j}\right)\right) && (29)\\
&= \sum_{\boldsymbol{\omega}'\in\mathcal{D}\left\{\mathcal{H}_k^{(\ell)},\mathcal{H}^{(\ell)}\right\}} \tilde{\tilde{b}}_{\boldsymbol{\omega}',j}\sin\left(\langle\boldsymbol{\omega}',\boldsymbol{r}\rangle+\tilde{\tilde{\tilde{\phi}}}_{\boldsymbol{\omega}',j}\right) && (30)
\end{aligned}$$

where last equality holds because of Lemma 2 and we have:

$$\begin{aligned}
\mathcal{D}\left\{\mathcal{H}_k^{(\ell)},\mathcal{H}^{(\ell)}\right\} &= \left\{\boldsymbol{\omega}_1\pm\boldsymbol{\omega}_2\,\middle|\,\boldsymbol{\omega}_1\in\mathcal{H}_k^{(\ell)},\boldsymbol{\omega}_2\in\mathcal{H}^{(\ell)}\right\} && (31)\\
&\subseteq \left\{\boldsymbol{\omega}_1\pm\boldsymbol{\omega}_2\,\middle|\,\boldsymbol{\omega}_1\in\tilde{\mathcal{H}}_{kK^\ell},\boldsymbol{\omega}_2\in\tilde{\mathcal{H}}_{K^\ell}\right\} && (32)\\
&= \left\{\sum_{t=0}^{T-1} s_t^{(1)}\boldsymbol{\Omega}_t \pm \sum_{t=0}^{T-1} s_t^{(2)}\boldsymbol{\Omega}_t\,\middle|\,\sum_{t=0}^{T-1}|s_t^{(1)}|\le kK^\ell,\sum_{t=0}^{T-1}|s_t^{(2)}|\le K^\ell\right\} && (33)\\
&= \left\{\sum_{t=0}^{T-1}\left(s_t^{(1)}\pm s_t^{(2)}\right)\boldsymbol{\Omega}_t\,\middle|\,\sum_{t=0}^{T-1}|s_t^{(1)}|\le kK^\ell,\sum_{t=0}^{T-1}|s_t^{(2)}|\le K^\ell\right\} && (34)\\
&\subseteq \left\{\sum_{t=0}^{T-1} s_t'\boldsymbol{\Omega}_t\,\middle|\,\sum_{t=0}^{T-1}|s_t'|\le (k+1)K^\ell\right\} = \tilde{\mathcal{H}}_{(k+1)K^\ell} && (35)\\
& && (36)
\end{aligned}$$

where the last line follows from triangle inequality. This proves our intermediate result in Eq. (27).

Now, let us use this result to complete the proof of the inductive step. In particular, we can now write Eq. (26) as

$$\begin{aligned}
\boldsymbol{z}_j^{(\ell+1)} &= \sum_{k=0}^{K}\alpha_k \sum_{\boldsymbol{\omega}'\in\mathcal{H}_k^{(\ell)}(\boldsymbol{\Omega})} \tilde{b}_{\boldsymbol{\omega}',j,k}\sin\left(\langle\boldsymbol{\omega}',\boldsymbol{r}\rangle+\tilde{\tilde{\phi}}_{\boldsymbol{\omega}',j,k}\right) && (37)\\
&= \sum_{\boldsymbol{\omega}'\in\mathcal{H}^{(\ell+1)}(\Omega)} c_{\boldsymbol{\omega}',j}\sin\left(\langle\boldsymbol{\omega}',\boldsymbol{r}\rangle+\phi_{\boldsymbol{\omega}',j}\right) && (38)
\end{aligned}$$

where $\mathcal{H}^{(\ell+1)} := \bigcup_{k=1}^{K}\mathcal{H}_k^{(\ell)} \subseteq \bigcup_{k=1}^{K}\tilde{\mathcal{H}}_{kK^\ell} \subseteq \tilde{\mathcal{H}}_{KK^\ell} = \tilde{\mathcal{H}}_{K^{\ell+1}}$. This sequence of inclusions concludes the proof.

□

S1.2. Two-layer SIREN example

Example. *Let $f_{\boldsymbol{\theta}}$ be a three-layer SIREN defined as* [1]

$$f_{\boldsymbol{\theta}}(r) = \boldsymbol{w}^{(2)^\top} \sin\left(\boldsymbol{W}^{(1)} \sin\left(\boldsymbol{\Omega} r\right)\right), \tag{39}$$

where $r \in \mathbb{R}$, $\boldsymbol{\Omega} \in \mathbb{R}^T$, $\boldsymbol{W}^{(1)} \in \mathbb{R}^{F\times T}$, and $\boldsymbol{w}^{(2)} \in \mathbb{R}^F$. The output of this network can equivalently be represented as

$$f_{\boldsymbol{\theta}}(r) = \sum_{m=0}^{F-1} \sum_{s_1,\ldots,s_T=-\infty}^{\infty} c_{m,s_1,\ldots,s_T} \sin\left(\left(\sum_{t=0}^{T-1} s_t\omega_t\right) r\right), \tag{40}$$

where $\omega_t^\top \in \mathbb{R}^D$ denotes the t^{th} row of Ω,

$$c_{m,s_1,\ldots,s_T} = \left(\prod_{t=0}^{T-1} J_{s_t}\left(W_{m,t}^{(1)}\right)\right) w_m^{(2)}, \tag{41}$$

and J_s denotes the Bessel function of first kind of order s.

Proof. As we have discussed before, the first layer in SIREN plays the role of the frequency mapping, i.e.

$$\boldsymbol{z}^{(0)} = \sin\left(\boldsymbol{W}^{(0)}\boldsymbol{r}\right) = \sin(\boldsymbol{\Omega}\boldsymbol{r}). \tag{42}$$

Hence the input of a node at the next layer is a linear combination of sinusoids at mapping frequencies. The output of a node at second layer can be written as:

$$\begin{aligned}
z_m^{(1)} &= \sin\left(\boldsymbol{W}_{m,:}^{(1)} \sin\left(\boldsymbol{\Omega} r\right)\right) && (43)\\
&= \sin\left(\sum_{t=0}^{T-1} \boldsymbol{W}_{m,t}^{(1)} \sin\left(\omega_t r\right)\right) && (44)\\
&= \operatorname{Im}\left\{\exp\left(j\left(\sum_{t=0}^{T-1} \boldsymbol{W}_{m,t}^{(1)} \sin\left(\omega_t r\right)\right)\right)\right\} && (45)\\
&= \operatorname{Im}\left\{\prod_{t=0}^{T-1} \exp\left(j\boldsymbol{W}_{m,t}^{(1)} \sin\left(\omega_t r\right)\right)\right\} && (46)\\
&= \operatorname{Im}\left\{\prod_{t=0}^{T-1} \sum_{s_t=-\infty}^{\infty} J_{s_t}(\boldsymbol{W}_{m,t}^{(1)}) \exp\left(js_t\omega_t r\right)\right\} && (47)\\
&= \operatorname{Im}\left\{\sum_{s_0=-\infty}^{\infty} \cdots \sum_{s_{T-1}=-\infty}^{\infty} \prod_{t=0}^{T-1} J_{s_t}(\boldsymbol{W}_{m,t}^{(1)}) \exp\left(js_t\omega_t r\right)\right\} && (48)\\
&= \sum_{s_1,\ldots,s_T=-\infty}^{\infty} \operatorname{Im}\left\{\prod_{t=0}^{T-1} J_{s_t}(\boldsymbol{W}_{m,t}^{(1)}) \exp\left(js_t\omega_t r\right)\right\} && (49)\\
&= \sum_{s_1,\ldots,s_T=-\infty}^{\infty} \operatorname{Im}\left\{\left(\prod_{t=0}^{T-1} J_{s_t}(\boldsymbol{W}_{m,t}^{(1)})\right) \exp\left(j\sum_{t=0}^{T-1} s_t\omega_t r\right)\right\} && (50)\\
&= \sum_{s_1,\ldots,s_T=-\infty}^{\infty} \left(\prod_{t=0}^{T-1} J_{s_t}(\boldsymbol{W}_{m,t}^{(1)})\right) \operatorname{Im}\left\{\exp\left(j\sum_{t=0}^{T-1} s_t\omega_t r\right)\right\} && (51)\\
&= \sum_{s_1,\ldots,s_T=-\infty}^{\infty} \left(\prod_{t=0}^{T-1} J_{s_t}(\boldsymbol{W}_{m,t}^{(1)})\right) \sin\left(\sum_{t=0}^{T-1} s_t\omega_t r\right) && (52)
\end{aligned}$$

[1]Note that we have omitted the bias terms to simplify the notation. These biases only change the phase terms in the sinusoids of the sum.

where $J_n(\cdot)$ represents the Bessel function of the first kind of order n and (47) follows from the Fourier series expansion of $\exp(j\beta \sin(\omega_0 x))$:

$$\exp(j\beta \sin(\omega_0 x)) = \sum_{n=-\infty}^{\infty} J_n(\beta) \exp(jn\omega_0 x). \tag{53}$$

Therefore, the output of the neural network can be written as:

$$f_{\boldsymbol{\theta}}(r) = \boldsymbol{w}^{(2)\top} z^{(1)} = \sum_{m=0}^{F-1} \boldsymbol{w}_m^{(2)} z_m^{(1)} = \sum_{m=0}^{F-1} \boldsymbol{w}_m^{(2)} \sum_{s_1,\ldots,s_T=-\infty}^{\infty} \left(\prod_{t=0}^{T-1} J_{s_t}(\boldsymbol{W}_{m,t}^{(1)}) \right) \sin\left(\sum_{t=0}^{T-1} s_t \omega_t r \right) \tag{54}$$

$$= \sum_{m=0}^{F-1} \sum_{s_1,\ldots,s_T=-\infty}^{\infty} \boldsymbol{w}_m^{(2)} \left(\prod_{t=0}^{T-1} J_{s_t}(\boldsymbol{W}_{m,t}^{(1)}) \right) \sin\left(\sum_{t=0}^{T-1} s_t \omega_t r \right) \tag{55}$$

□

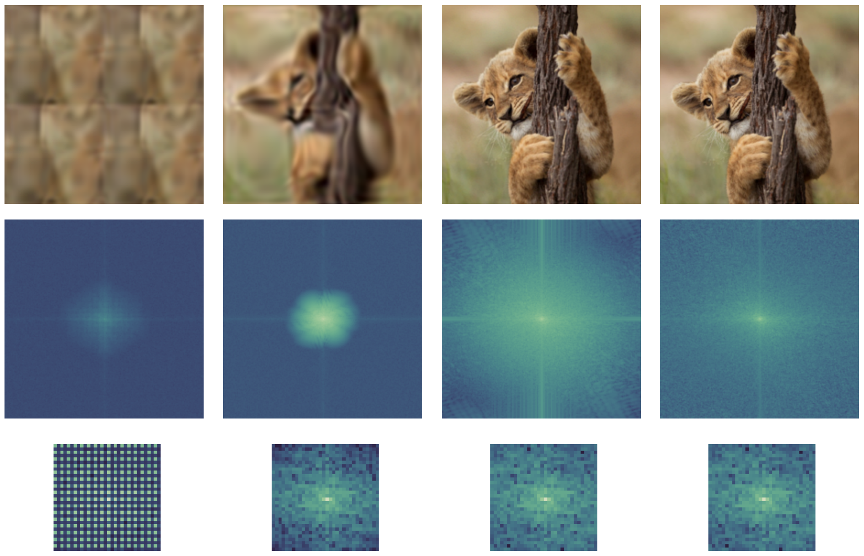
S2. Imperfect recovery

S2.1. Experimental details

For the experiment in Section 4.1, we train an FFN with four fully connected layers: three hidden layers with dimension 256 followed by an output layer of dimension $C = 3$. All the networks presented in Fig.2 are trained for 2000 iterations with Adam optimizer [2]. The training image has the size of 512×512 and we use all the available pixels during training[2].

S2.2. Additional experiments

We further demonstrate the imperfect recovery phenomenon with INRs for different networks and configurations. In Fig. S1, we present the results for SIREN [7], where the first layer of the form $\boldsymbol{z}^{(0)} = \sin\left(\omega_0(\boldsymbol{W}^{(0)}\boldsymbol{r} + \boldsymbol{b}^{(0)})\right)$ can be considered as the input mapping $\gamma(\boldsymbol{r})$. For a fair comparison with FFNs and to better illustrate the strong dependence of the learned representation on the chosen mapping, we do not perform any updates on the parameters of the initial layer, i.e., $\boldsymbol{W}^{(0)}$ and $\boldsymbol{b}^{(0)}$, during training. The initialization of these parameters ensures a similar mapping to that of FFN presented in Figure 2 of the main text, i.e. two single frequency mappings with frequencies $f_0 = 1$ and $f_0 = 0.5$ followed by a rich mapping. As for the rest of the architecture, we use the same number of layers and training strategy. As you can see in Fig. S1, initializing the SIREN with this $\gamma(\boldsymbol{r})$ results also in a set $\mathcal{H}(\boldsymbol{\Omega}) \subseteq \{2\pi k | k \in \mathbb{Z}\}$, which leads to a very imperfect recovery with the reconstruction looking aliased in the spatial domain.



(a) SIREN ($\omega_0 = 1$) $\boldsymbol{W}^{(0)} = I$ (b) SIREN ($\omega_0 = 1$) $\boldsymbol{W}^{(0)} = 0.5 \times I$ (c) SIREN ($\omega_0 = 30$) $\boldsymbol{W}^{(0)} \in \mathbb{R}^{256\times 2}$ (d) Ground Truth

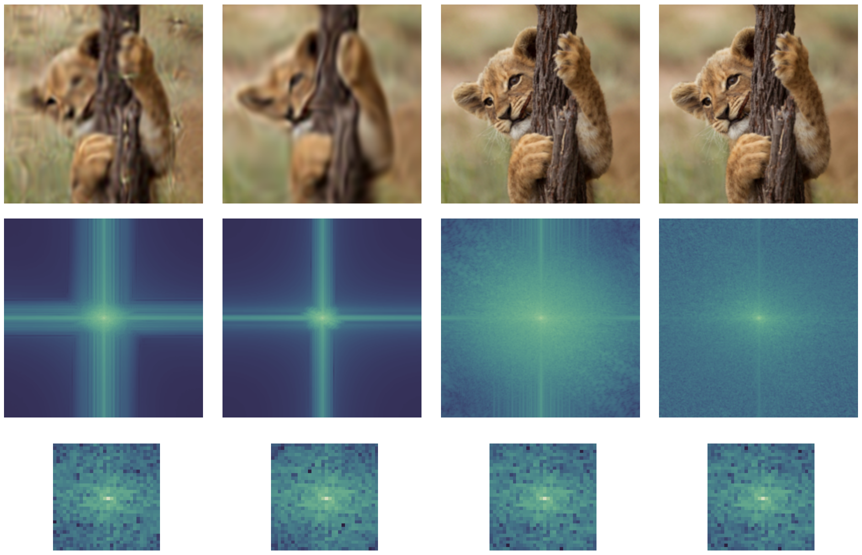
Figure S1. **Top row:** Image reconstruction with SIREN [7] for different configurations. **Middle row:** Magnitude of the DFT of the reconstruction. **Bottom row:** The center crop of size 32×32 from the magnitude of the DFT of the reconstruction.

Note, however, that in the original formulation of SIREN [7], the parameters of the initial layer are also updated during training. To see the effect that this has in the reconstruction, we repeat the same experiment with a SIREN[3] with trainable $\boldsymbol{W}^{(0)}$ and $\boldsymbol{b}^{(0)}$. Interestingly, as illustrated in Fig. S2, the aliasing effect is mostly eliminated in this case. This a result of having a dynamic input mapping, which enables spreading the energy over all spectral coefficients rather than only the even ones. Nevertheless, similar to our observations for Fig.2, we see that the reconstruction for low values of ω_0 is also blurry with most of the energy concentrated in the low frequencies.

For the sake of completeness, we also investigate the effect of having learnable parameters in $\gamma(\boldsymbol{r})$ for FFNs. Specifically, we repeat this same experiment for the single frequency mapping of the main text, $\gamma(\boldsymbol{r}) = [\cos(2\pi \boldsymbol{f}\boldsymbol{r}), \sin(2\pi \boldsymbol{f}\boldsymbol{r})]^T$, where we initialize the frequency variable f as f_0. Note that f, now, is a trainable parameter. We denote this network as SFM*.

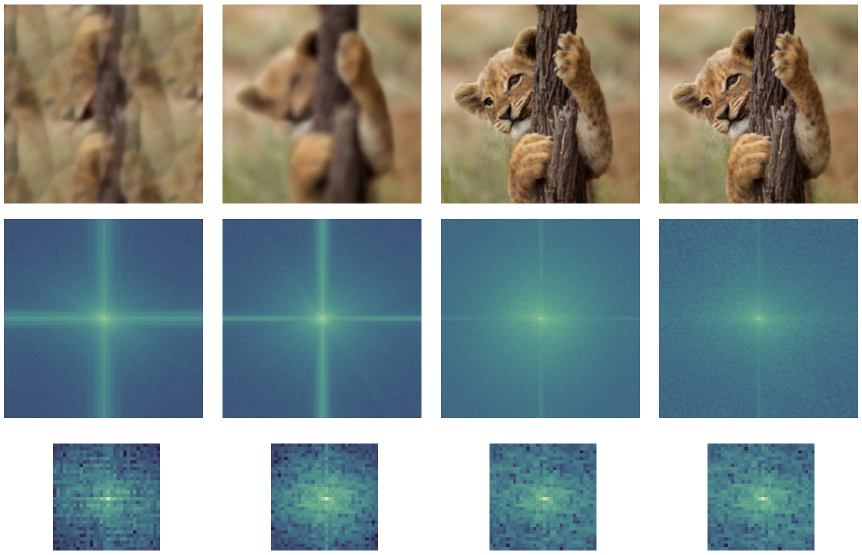
[2]The code to reproduce all our experiments using JAX [1] will be released upon acceptance.

[3]The notation *, e.g., SIREN*, denotes the corresponding INR involving input mapping with learnable parameters.



(a) SIREN* ($\omega_0 = 1$) $\boldsymbol{W}^{(0)} = I$ (b) SIREN* ($\omega_0 = 1$) $\boldsymbol{W}^{(0)} = 0.5 \times I$ (c) SIREN* ($\omega_0 = 30$) $\boldsymbol{W}^{(0)} \in \mathbb{R}^{256\times 2}$ (d) Ground Truth

Figure S2. **Top row:** Image reconstruction with SIREN [7] for different configurations. The notation * indicates that the parameters of the initial layer are subject to gradient updates during training. **Middle row:** Magnitude of the DFT of the reconstruction. **Bottom row:** The center crop of size 32×32 from the magnitude of the DFT of the reconstruction.



(a) SFM* ($f_0 = 1$) (b) SFM* ($f_0 = 0.5$) (c) FFN* ($\sigma = 1$) (d) Ground Truth

Figure S3. **Top row:** Image reconstruction with FFN [9] for different configurations. SFM stands for single frequency mapping as in Figure 2, i.e., $\gamma(\boldsymbol{r}, f) = [\cos(2\pi f \boldsymbol{r}), \sin(2\pi f \boldsymbol{r})]^T$ where the frequency variable is initialized as f_0 and subject to gradient updates. In general, the notation * indicates learnable input mapping. **Middle row:** Magnitude of the DFT of the reconstruction. **Bottom row:** The center crop of size 32×32 from the magnitude of the DFT of the reconstruction.

Similarly, we also train with an FFN* with trainable $\boldsymbol{\Omega}_{i,j}$, randomly initialized as explained in Sec.2. Figure S3 shows the results of these experiments with identical findings as in the main text and the SIRENs presented above.

S3. Aliasing

S3.1. Experimental details

For the experiment in Section 4.2, we train a SIREN with three fully connected layers: two hidden layers with dimension 128 followed by the output layer of dimension 1. All the networks presented in Fig.3 are trained for 2000 iterations using Adam [2] with a learning rate of 1×10^{-4}. For the training data, we generate a single frequency signal $g(r) = \sin(2\pi \cdot 23r)$ on 128 evenly spaced samples in the range $[0, 1]$, i.e., sampled with a frequency of $f_s = 128$ and test the learned representation $f_\theta(r)$ with samples from the same signal with $f_s = 256$.

S3.2. Additional experiments

For the sake of completeness, and to show that aliasing is prevalent across INR families, we repeat the same experiment using FFNs [9] initialized with different σ. Recall that σ determines the standard deviation of the the distibution of $\mathbf{\Omega}_{i,j}$ at initialization, i.e., $\mathbf{\Omega}_{i,j} \sim \mathcal{N}(0, \sigma^2)$. The results presented in Fig. S4 highlight that the same aliasing phenomenon observed for SIRENs in Fig.3 happens identically for FFNs.

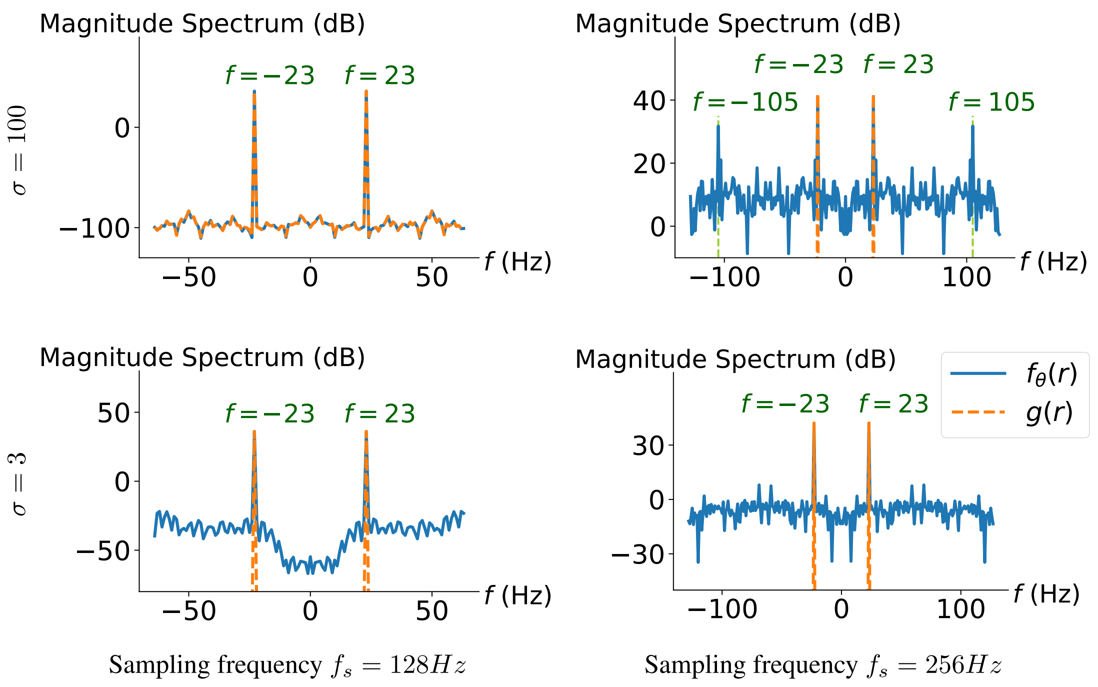


Figure S4. Magnitude of the spectrum of $g(r) = \sin(2\pi \cdot 23r)$ and its FFN [9] reconstruction trained at $f_s = 128$ Hz. **Top row** shows $\sigma = 100$, and **bottom row** $\sigma = 3$. On the **left** the signals are sampled at $f_s = 128$ Hz and on the **right** at $f_s = 256$ Hz.

S4. NTK eigenfunctions as dictionary atoms

S4.1. Estimation of eigenfunctions of the NTK

As it is common in the kernel literature, in this work, we use the eigenvectors of the kernel Gram matrix to approximate the eigenfunctions of the NTK. Specifically, unless stated otherwise, in all our experiments we compute the Gram matrix of the NTK at any $\boldsymbol{\theta}_0$ using as samples the coordinates of all the pixels of an image, laid out on a grid of fixed resolution (64×64). That is, we compute a Gram matrix of size $64^2 \times 64^2$. To that end, we use the `empirical_kernel_fn` function from the `neural_tangents` library [5] which allows to compute this matrix using a batch implementation. Note that this operation can be computationally intense, scaling quadratically with the number of samples, but also quadratically with the number of outputs. For this reason, we decided to use INRs with a single output and work only with grayscale images. The results, however, are easily extensible to the multi-output setting.

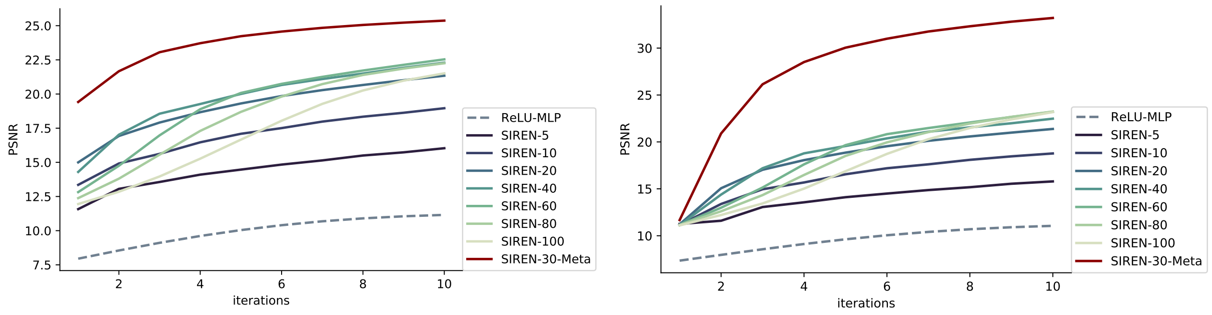
Once we have the Gram matrix, we can perform its eigendecomposition and use the resulting eigenvectors to approximate the values of the eigenfunctions $\phi_j(\boldsymbol{r})$ at the pixel coordinates. The inner products $\langle \phi_j, g_n \rangle$ are then easily approximated as

$$\langle \phi_j, g_n \rangle \approx \sum_{i=1}^{64^2} \phi_j(\boldsymbol{r}_i) g_n(\boldsymbol{r}_i). \tag{56}$$

S4.2. Training details

In Section 5.2, we compare the generalization performance of several SIRENs (four hidden layers with dimension 256 followed by the output layer of dimension 1) with different initialization strategies. To that end, we train each of these networks to reconstruct 100 validation images from the CelebA dataset using half of the pixels of the images for training. The training pixel locations are random, but we use the same across all validation images. Generalization performance, is then tested using the remaining half of the pixels. To be consistent with the empirical protocol proposed in [8], we use full batch Adam [2] with a learning rate of 10^{-4} for the randomly initialized weights and use full batch gradient descent with learning rate 10^{-2} for the meta learned weights, which they reported to be the optimal choice of optimizers and learning rates for each individual case.

Figure S5 shows the evolution of the average training and test curves for each of these networks. As we can see, the networks with a better energy concentration in Fig. 4, are the networks that train faster, and reach the best test performances.



(a) Test (b) Training

Figure S5. The evolution of the reconstruction performance (average PSNR) of different representations for the first 10 training iterations when trained on 100 validation images from CelebA dataset. The numbers for different SIREN instances indicate the value of ω_0. Meta indicates the learned initialization, whereas all the other networks are initialized randomly in accordance with the original implementation [7].

S4.3. Experiments on additional networks

For completeness, we also provide the results of these experiments using FFNs (a sinusoidal mapping of size 256 followed by three hidden layers with dimension 256 followed by the output layer of dimension 1), instead of SIRENs in Fig. S6. Again, we observe that those networks which have an energy profile more concentrated on the largest eigenvalues perform much better.

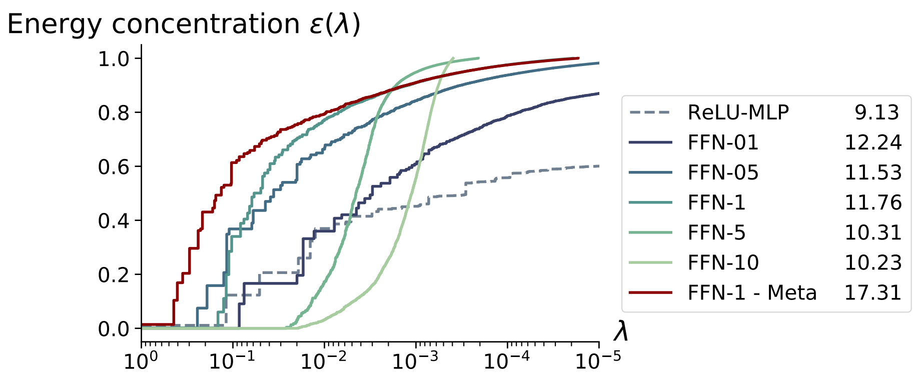


Figure S6. Average energy concentration of 100 validation images from CelebA on subspaces spanned by the eigenfunctions of the empirical NTK associated to eigenvalues greater than a given threshold. Legend shows the average test PSNR after training to reconstruct those images from 50% randomly selected pixels for 3 iterations. The value following FFN specifies the σ parameter of the given network.

S5. Meta-learning experiment

S5.1. Experimental details

Our meta-learning experiments consist of two phases: A first pre-training phase, in which we use MAML, to meta-learn a good initialization from $5,000$ training images from CelebA using a learning rate of 10^{-5} as indicated in [8] and 5000 meta-iterations. In our experiments, we use a SIREN with four hidden layers with dimension 256 followed by the output layer of dimension 1, randomly initialized prior to meta-learning using $\omega_0 = 30$. After pretraining, we finetune the networks starting at the meta-learned weights using the training protocol described in Sec. S4.2.

To estimate the eigenfunctions of the NTK at the meta-learned weights we use the experimental setting described in Sec. S5.1.

S5.2. Experiments with an additional meta-learning algorithm

We repeat the same experiments by replacing the meta-learning algorithm with Reptile. Fig. S7 shows the resulting energy concentration plot and Fig. S11 shows the eigenfunctions of the NTK at the meta-learned weights using Reptile. As we can see, the results agree completely with those found using MAML. This suggests that the reshaping effect of meta-learning on the NTK is a general phenomenon, which might be induced using multiple algorithms.

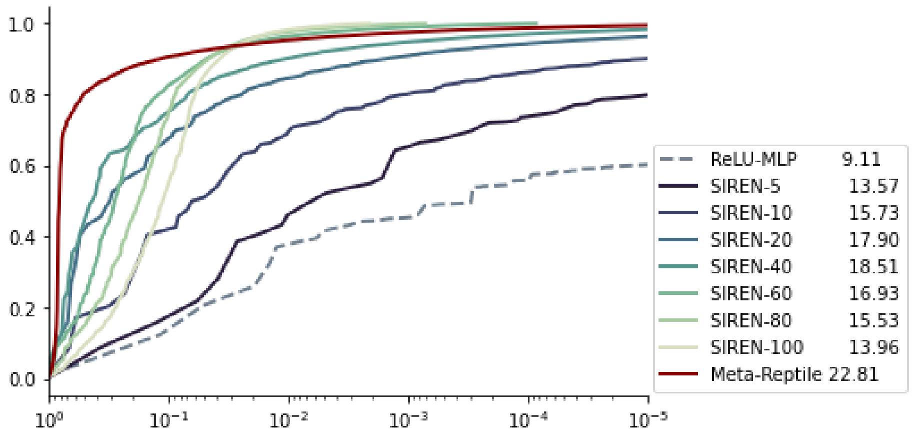


Figure S7. Average energy concentration of 100 validation images from CelebA on subspaces spanned by the eigenfunctions of the empirical NTK associated to eigenvalues greater than a given threshold. Legend shows the average test PSNR after training to reconstruct those images from 50% randomly selected pixels. The meta-learned weights are computed using Reptile.

S5.3. Experiments on additional networks

For completeness, we also provide the results of these experiments using an FFN (a sinusoidal mapping of size 256 followed by three hidden layers with dimension 256 followed by the output layer of dimension 1) instead of a SIREN. Prior to meta-learning the input mapping is initialized using $\sigma = 1$. Fig. S6 shows these additional results, where we see that meta-learning does also improve the energy concentration of the validation images on the principal eigenspace of the NTK for the FFNs. Performance does also improve significantly in this case.

S5.4. Comparison with standard training

Finally, as a baseline, we provide a comparison between standard training and meta-learning, both in terms of the dynamics of the NTK, and fine-tuning performance. Specifically, we repeated the same experimental pipeline described before, where instead of pretraining using MAML, we pretrain a SIREN using Adam to reconstruct one training image in CelebA.

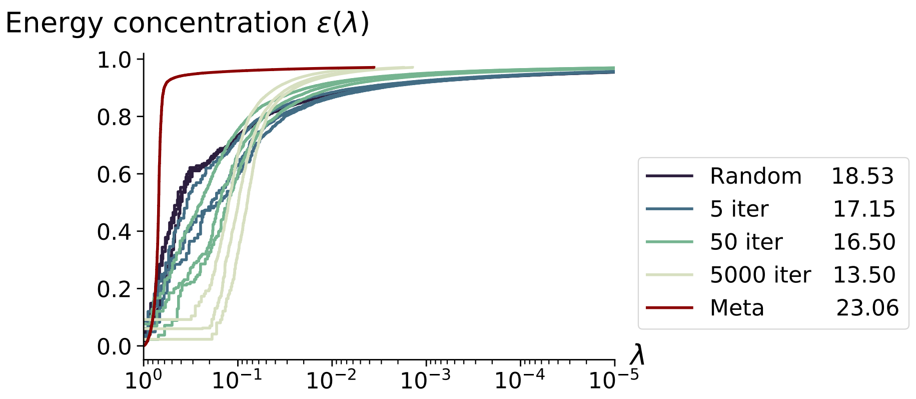
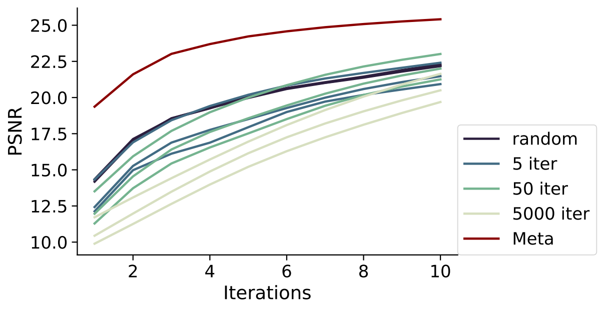
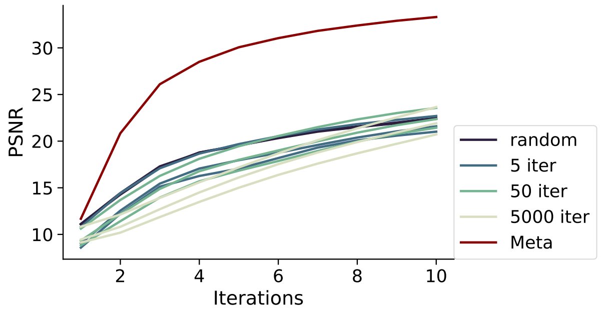


Figure S8. Average energy concentration of 100 validation images from CelebA on subspaces spanned by the eigenfunctions of the empirical NTK associated to eigenvalues greater than a given threshold. Legend shows the average test PSNR after training to reconstruct those images from 50% randomly selected pixels. The number of iterations specify those of pretraining on the single task.

As we can see in Fig. S8 and Fig. S11 has a signficant impact on the NTK. As described before in [3], training a neural network to reconstruct a signal transforms the eigenfunctions of the final NTK such that they look like the target signals. Surprisingly, though, we are seeing that for this particular set of tasks, fine-tuning from those initializations results in worse performance than starting from scratch, i.e., there is a negative transfer between different tasks.



(a) Test



(b) Training

Figure S9. The evolution of the reconstruction performance (PSNR) of a SIREN, where the parameters of the network are (i) initialized randomly, (ii) pretrained on a single image for different number of training iterations, and (iii) meta-learned. Note that three different realizations of the networks with pretrained weights on a single image are provided for each different number of training iterations.

We can understand this phenomenon using the signal dictionary analogy, which is summarized by Fig. S8. As we can see, the more we pretrain the networks to reconstruct a specific training image, the more the energy profile of the validation set shifts to the smallest eigenvalues of the pretrained kernel, i.e, the learned dictionary has a worse compression performance than the randomly initialized NTK. As a result, as illustrated in Fig. S9, we see that the more pretrained networks take longer to converge when fine-tuned, and reach worse generalization performances.

Inspecting the eigenfunctions of these different networks, and interpreting them as dictionary atoms, gives us one last piece of intuition to understand the dynamics of meta-learning. Indeed, as we can see in Fig. S11 the eigenfunctions of the standardly trained networks do really look like the target signals, having very specific crisp edges and textures, which might not directly appear on other images. That is, the dictionaries of these networks are overfitted to their specific pretraining task. On the other hand, meta-learning can leverage much more data to construct a dictionary whose atoms can be easily composed to create face images. We believe that understanding the dynamics of this process will be a very fruitful avenue for future research.

S6. Performance on NTK eigenfunctions

In this section we empirically demonstrate that it is easier for a network to learn the NTK eigenfunctions corresponding to larger eigenvalues. Figure S10 shows the PSNR reached after training the network for 2000 iterations with the target set to the NTK eigenfunction at initialization with the corresponding index. Even though the meta-learned weights are used in this experiment, this phenomenon applies to other networks and weights as well and is previously discussed in [6].

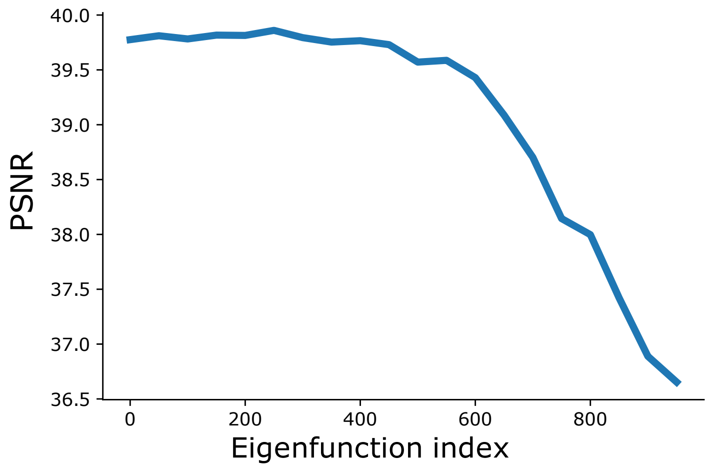


Figure S10. PSNR performance of a SIREN trained on increasing eigenfunctions of its NTK at initialization.

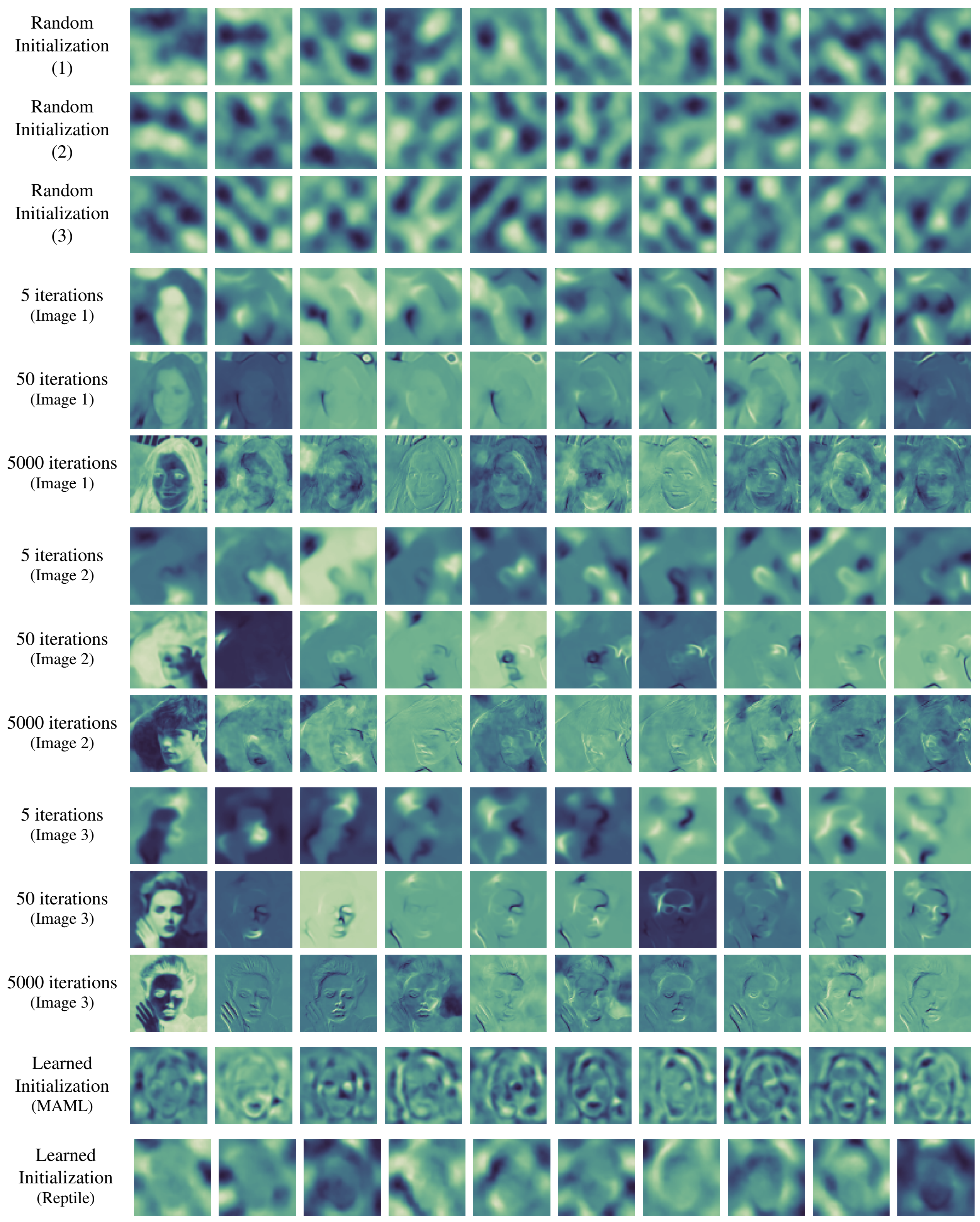


Figure S11. First ten eigenfunctions corresponding to the largest eigenvalues of the empirical NTK of SIREN [7] at initialization for different cases. The first three rows represent different realizations of random initialization. Rows 4-5-6 illustrate the eigenfunctions of a SIREN after meta-learning on a randomly chosen (single) training image from the CelebA dataset [4] with corresponding number of meta-learning updates on learnable parameters [8]. Rows 7-8-9 and 10-11-12 are the outcomes of the same experiment for different choice of the training image used for meta-learning. The bottom row depicts the eigenfunctions of the learned initialization when we use $5,000$ training images from the CelebA dataset for meta-learning.

References

[1] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. S8

[2] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. S8, S10, S11

[3] Dmitry Kopitkov and Vadim Indelman. Neural spectrum alignment: Empirical study. In *International Conference on Artificial Neural Networks (ICANN)*, 2020. S13

[4] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015. S15

[5] Roman Novak, Lechao Xiao, Jiri Hron, Jaehoon Lee, Alexander A. Alemi, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. Neural tangents: Fast and easy infinite neural networks in python. In *International Conference on Learning Representations (ICLR)*, 2020. S11

[6] Guillermo Ortiz-Jiménez, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. What can linearized neural networks actually say about generalization? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. S14

[7] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. S8, S9, S11, S15

[8] Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P. Srinivasan, Jonathan T. Barron, and Ren Ng. Learned initializations for optimizing coordinate-based neural representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. S11, S12, S15

[9] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. S9, S10