Seeing Implicit Neural Representations as Fourier Series

Nuri Benbarka*, Timon Höfer*, Hamd ul-moqeet Riaz, Andreas Zell
University of Tübingen, Wilhelm-Schickard-Institute for Computer Science, Sand 1, 72076, Tübingen
firstname.lastname@uni-tuebingen.de

Abstract

Implicit Neural Representations (INR) use multilayer perceptrons to represent high-frequency functions in low-dimensional problem domains. Recently these representations achieved state-of-the-art results on tasks related to complex 3D objects and scenes. A core problem is the representation of highly detailed signals, which is tackled using networks with periodic activation functions (SIRENs) or applying Fourier mappings to the input. This work analyzes the connection between the two methods and shows that a Fourier mapped perceptron is structurally like one hidden layer SIREN. Furthermore, we identify the relationship between the previously proposed Fourier mapping and the general d-dimensional Fourier series, leading to an integer lattice mapping. Moreover, we modify a progressive training strategy to work on arbitrary Fourier mappings and show that it improves the generalization of the interpolation task. Lastly, we compare the different mappings on the image regression and novel view synthesis tasks. We confirm the previous finding that the main contributor to the mapping performance is the size of the embedding and standard deviation of its elements.

1. Introduction

INR is a novel field of research in which the traditional discrete signal representation (e.g., images as discrete grids of pixels, 3D shapes as voxel grids or meshes) are replaced with continuous functions that map the input domain of the signal (e.g., coordinates of a specific pixel in the image) to a representation of color, occupancy or density at the input location. However, these functions typically are not analytically tractable; INRs approximate those functions with fully connected neural networks.

INRs are not coupled to the spatial resolution (e.g., voxel size in a 3D scene) and theoretically have infinite resolution. Therefore, these representations are naturally suited in applications with high-dimensional signals and heavy memory consumption. Also, since they are differentiable, they are suitable for gradient-based optimization and machine learning. In addition, the application of INRs for images [10, 31], volume density [19], and occupancy [17] enhanced the performance on various tasks such as shape representation [4, 5, 8, 7, 12, 18, 25], texture synthesis [10, 22], and shape inference from images [16, 15].

*equal contribution

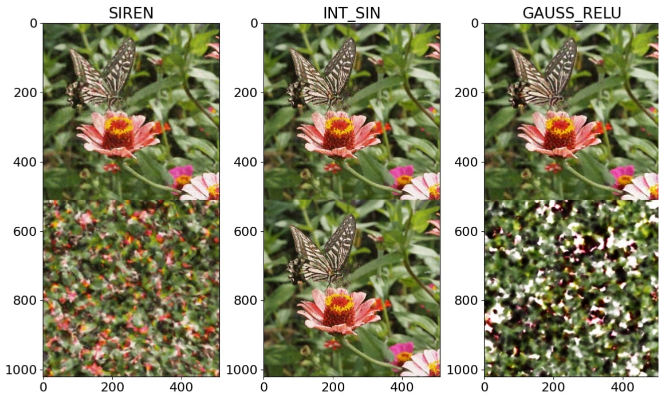


Figure 1: Visualization of SIREN [29], Gauss ReLU [32] and our method (INT_SIN). The top row is the first period and bottom row is the second period, which shows our method enforcing periodicity.

However, early architectures lacked accuracy in high-frequency details. Sitzmann et al. [29] proposed SIRENs, which could represent high frequencies. They argued that sinusoidal activations work better than ReLU networks because ReLU networks are piecewise linear, and their second derivative is zero. As a result, they are incapable of modeling data contained in higher-order derivatives of signals. However, a concurrent work [19] proposed positional encoding, which also enabled the networks to learn high-frequency information. The positional encoding uses a heuristic sinusoidal mapping to input coordinates before passing them through a ReLU network. They did follow-up work [32] exploring the general Fourier mapping and explaining why it worked using a Neural Tangent Kernel

(NTK) framework [11]. They found out that the Fourier mapping transforms the NTK into a shift-invariant kernel. And modifying the mapping parameters enables tuning the NTK's spectrum, therefore controlling the range of frequencies the network can learn. They also showed that a random Fourier mapping with low standard deviation learns only low frequencies of the signal. On the contrary, a high standard deviation lets the network learn high frequencies only, which leads to over-fitting. They recommended a linear search to find the optimal value of the standard deviation for the corresponding task. They also showed that increasing the number of parameters in the mapping improves the performance constantly.

However, we wonder what the difference between SIRENs and Fourier mapping is? Will the performance be saturated when we continue to increase the mapping parameters? Is there a way to avoid over-fitting when training networks using Fourier mapping? And is random Fourier mapping the optimal mapping?

To answer these questions, we explored the mathematical connection between Fourier mappings and SIRENs and showed that a Fourier mapped perceptron is structurally like a one hidden layer SIREN. However, in the SIREN case, the mapping is trainable, and it is represented in the amplitude-phase form instead of the sine-cosine form in the case of Fourier mappings.

Also, we looked at the functions we want to learn, and we observed that they have a limited input domain (e.g., the height and width of an image), and their values are defined on a finite set. Hence, we can assume that they are continuous and periodic over their input bound, which satisfies all the requirements to represent them with a Fourier series. Furthermore, we determined the d-dimensional Fourier series's trigonometric form and showed that it is precisely a single perceptron with an integer lattice mapping applied to its inputs. The weights of that perceptron are the Fourier series coefficients. As the Fourier series can theoretically represent any periodic signal, this perceptron can represent any periodic signal if it has an infinite number of frequencies in its mapping. However, in practice, the Fourier series coefficients are finite, and we can get them by sampling the signal at the Nyquist rate (twice the bandwidth) and applying a fast Fourier transform (FFT) to the signal. Thus, the number of Fourier coefficients is the theoretical upper bound of the number of parameters needed in the mapping.

Moreover, we modified the progressive training strategy of [13], where we train the lower frequencies in the initial training phase and gradually add the higher frequency components as the training progresses. As a result, we show that our *Progressive Training* strategy avoids the problem of over-fitting. Finally, we tested our proposed *Integer Lattice* mapping in the image regression and novel view synthesis tasks. We found out that the main contributor to the mapping performance is the number of parameters and the standard deviation, as was shown in [32]. In summary, we offer the following contributions:

- We introduce an integer Fourier mapping and prove that a perceptron with this mapping is equivalent to a Fourier series.
- We explore the mathematical connection between Fourier mappings and SIRENs and show that a Fourier mapped perceptron is structurally like a one hidden layer SIREN.
- We show that the integer mapping forces periodicity of the network output.
- We modify the progressive training strategy of [13] and show that it improves the generalization of the interpolation task.
- We compare the different mappings on the image regression and novel view synthesis tasks and verify the previous findings of [32] that the main contributor to the mapping performance is the number of elements and standard deviation.

2. Related work

Inspired by INRs' recent success, by outperforming grid-, point- and mesh-based representations (for the first time in 2018 [25],[17],[3]), many works based on INRs achieved state-of-the-art results in 3D computer vision [1, 9, 12, 26, 2, 29]. Moreover, impressive results are obtained across different input domains, e.g., from 2D supervision [30, 21, 19], 3D supervision [27, 23], to dynamic scenes [20] which can be represented by space-time INR.

In early architectures, there was a lack of accuracy in fine details of signals. Mildenhall et al. [19] proposed positional encodings to tackle this problem, then Tancik et al. [32] further explored positional encodings in an NTK framework, showing that mapping input coordinates to a representation close to the actual Fourier representation before passing them to the MLP lead to a good representation of the high-frequency details. Furthermore, they showed that random Fourier mappings achieved superior results than if one takes the simple positional encoding. Sitzmann et al. [29] also attempted to solve the problem of getting high-frequency details. They proposed SIRENs and demonstrated that SIRENs are suited for representing complex signals and their derivatives. In both solutions, they used a variant of Fourier neural networks (FNN) for the first layer of the MLP. FNN are neural networks that use either sine or cosine activations to get their features [14].

The first attempt to build an FNN was by [6]. They proposed a one-layer hidden neural network with a cosine squasher activation function and showed if they hand-wire certain weights, it will represent a Fourier series. Silvescu [28] proposed a network that did not resemble a standard feedforward neural network. However, they used a cosine activation function to get the features. Liu et al. [14] introduced the general form for Fourier neural networks in a feedforward manner. They also proposed a strategy to initialize the frequencies of the embedding, which helped for convergence. Our work will show another way to initialize the embedding, which results in a neural network that is precisely a Fourier series.

3. Method

3.1. Integer lattice mapping

This section explains how a perceptron with an integer lattice Fourier mapping applied to its inputs is equivalent to a Fourier series. First, we present the Fourier mapped perceptron equation and then link it to the Fourier series's general equation. The fundamental building block of any neural network is the perceptron, and it is defined as

$$\mathbf{y}(\mathbf{x}, \mathbf{W}', \mathbf{b}) = g(\mathbf{W}' \cdot \mathbf{x} + \mathbf{b}). \tag{1}$$

Here $\mathbf{y} \in \mathbb{R}^{d_{out}}$ is the perceptron's output, $g(\cdot)$ is the activation function (usually non-linear), $\mathbf{x} \in \mathbb{R}^{d_{in}}$ is the input, $\mathbf{W}' \in \mathbb{R}^{d_{out} \times d_{in}}$ is the weight matrix, and $\mathbf{b} \in \mathbb{R}^{d_{out}}$ is the bias vector. Now, if we let $g(\cdot)$ to be the identity function and apply a Fourier mapping to the input we get

$$\mathbf{y}(\mathbf{x}, \mathbf{W}) = \mathbf{W} \cdot \gamma(\mathbf{x}) + \mathbf{b}, \tag{2}$$

where $\gamma(x)$ is the Fourier mapping defined as

$$\gamma(\mathbf{x}) = \begin{pmatrix} \cos(2\pi \mathbf{B} \cdot \mathbf{x}) \\ \sin(2\pi \mathbf{B} \cdot \mathbf{x}) \end{pmatrix}. \tag{3}$$

$\mathbf{W} \in \mathbb{R}^{d_{out} \times 2m}$, $\mathbf{B} \in \mathbb{R}^{m \times d_{in}}$ is the Fourier mapping matrix, and m is the number of frequencies. Equation 2 is the general equation of a Fourier mapped perceptron, and we will relate it to the Fourier series's general equation.

A Fourier series is a weighted sum of sines and cosines with incrementally increasing frequencies that can reconstruct any periodic function when its number of terms goes to infinity. In applications that use coordinate-based MLPs, the functions we want to learn are not periodic. However, their inputs are naturally bounded (e.g., height and width of an image). Accordingly, it doesn't harm if we assume that the input is periodic over its input's bounds to represent it as a Fourier series. We will explain later why this assumption has many advantages. A function $f : \mathbb{R}^{d_{in}} \to \mathbb{R}^{d_{out}}$ is periodic with a period $p \in \mathbb{R}^{d_{in}}$ if

$$f(\mathbf{x} + \mathbf{n} \circ \mathbf{p}) = f(\mathbf{x}) \quad \forall \mathbf{n} \in \mathbb{Z}^d, \tag{4}$$

where $\circ$ is the Hadamard product. As it is plausible to normalize the inputs to their bounds, we assume that each variable's period is 1. The Fourier series expansion of function (4) with $\mathbf{p} = \mathbf{1}^d$ is defined by [24]:

$$f(\mathbf{x}) = \sum_{\mathbf{n} \in \mathbb{Z}^d} c_{\mathbf{n}} e^{2\pi i \mathbf{n} \cdot \mathbf{x}}, \tag{5}$$

where $c_{\mathbf{n}}$ are the Fourier series coefficients, and they are calculated by:

$$c_{\mathbf{n}} = \int_{[0,1]^d} f(\mathbf{x}) e^{-2\pi i \mathbf{n} \mathbf{x}} d\mathbf{x}. \tag{6}$$

For real-valued functions, it holds that $c_{\mathbf{n}} = c^*_{\mathbf{-n}}$ where $c^*_{\mathbf{n}}$ is the conjugate of $c_{\mathbf{n}}$. Using Euler's formula and mathematical induction we showed that equation (5) can be written as (see supplementary material):

$$f(\mathbf{x}) = \sum_{\mathbf{n} \in \mathbb{N}_0 \times \mathbb{Z}^{d-1}} a_{\mathbf{n}} \cos(2\pi \mathbf{n} \cdot \mathbf{x}) + b_{\mathbf{n}} \sin(2\pi \mathbf{n} \cdot \mathbf{x}) \tag{7}$$

$$\begin{aligned} a_{\mathbf{0}} &= c_{\mathbf{0}}, \\ a_{\mathbf{n}} &= \begin{cases} 0 & \exists j \in \{2, \dots, d\} : n_1 = \dots = n_{j-1} = 0 \wedge n_j < 0 \\ 2\text{Re}(c_{\mathbf{n}}) & \text{otherwise}, \end{cases} \\ b_{\mathbf{n}} &= \begin{cases} 0 & \exists j \in \{2, \dots, d\} : n_1 = \dots = n_{j-1} = 0 \wedge n_j < 0 \\ -2\text{Im}(c_{\mathbf{n}}) & \text{otherwise}. \end{cases} \end{aligned} \tag{8}$$

And if we write equation (7) in vector form, we get

$$f(\mathbf{x}) = (a_{\mathbf{B}}, b_{\mathbf{B}}) \cdot \begin{pmatrix} cos(2\pi \mathbf{B} \cdot \mathbf{x}) \\ sin(2\pi \mathbf{B} \cdot \mathbf{x}) \end{pmatrix}, \tag{9}$$

where $a_{\mathbf{B}} = (a_{\mathbf{n}})_{\mathbf{n} \in \mathbf{B}}$, and $b_{\mathbf{B}} = (b_{\mathbf{n}})_{\mathbf{n} \in \mathbf{B}}$. Now, if we compare 2 and 9, we find similarities. We see that $(a_{\mathbf{B}}, b_{\mathbf{B}})$ is equivalent to $\mathbf{W}$, $\mathbf{b}$ is zero and $\mathbf{B} = \mathbb{N}_0 \times \mathbb{Z}^{d-1}$, is the concatenation of all possible permutations of $\mathbf{n}$. For practicality we limit $\mathbf{B}$ to

$$\mathbf{B} = \{0, \dots, N\} \times \{-N, \dots, N\}^{d-1} \setminus \mathbf{H}, \tag{10}$$

where N will be called the frequency of the mapping, $\mathbf{H} = \{\mathbf{n} \in \mathbb{N}_0 \times \mathbb{Z}^{d-1} | \exists j \in \{2, \dots, d\} : n_1 = \dots = n_{j-1} = 0 \wedge n_j < 0\}$, then the perceptron represents a Fourier series. Hence, we calculate the dimension m of all possible permutations (see supplementary material)

$$m = (N+1)(2N+1)^{d-1} - \sum_{l=0}^{d-2} N(2N+1)^l. \tag{11}$$

In practice, we can find the Fourier series coefficients by sampling the function uniformly with a frequency higher than the Nyquist frequency and apply a Fast Fourier Transform (FFT) on the sampled signal. The resulting FFT coefficients are the Fourier series coefficients multiplied by the number of the sampled points. And in theory, if we initialize the weights with the Fourier series coefficients, our network should give the training target at iteration 0.

3.2. SIRENs and Fourier mapping comparison

In this section we want show that a Fourier mapped perceptron is structurally like a SIREN with one hidden layer. If we evaluate $\mathbf{W} \cdot \gamma(\mathbf{x})$ in equation (2), using (3) and combine the Sine and Cosine terms, we get:

$$\mathbf{y}(\mathbf{x}, \mathbf{W}) = \mathbf{W} \cdot sin(2\pi\mathbf{C} \cdot \mathbf{x} + \phi) + \mathbf{b}, \tag{12}$$

where $\phi := (\pi/2, \ldots, \pi/2, 0, \ldots, 0)^T \in \mathbb{R}^{2m}$ and $\mathbf{C} := (\mathbf{B}, \mathbf{B})^T$ (see supplementary material). Here we see that $\mathbf{C}$ is acting as the weight matrix applied to the input, ϕ is like the first bias vector and $sin(\cdot)$ is the activation function. Hence, the initial Fourier mapping can be represented by an extra initial SIREN layer, with the difference that $\mathbf{B}$ and ϕ are trainable in the SIREN case. This finding closes the bridge between Fourier frequency mappings and sinusoidal activation functions which have recently attracted a lot of attention.

3.3. Progressive training

Chen-Hsuan et al. [13] introduced a training strategy for coarse-to-fine registration for NeRFs which they called BARF. Their idea is to mask out the positional encoding's high-frequency activations at the start of training and gradually allow them during training. Their work showed how to use this strategy on positional encodings only to improve camera registration. In our work, we will show how to run this strategy on an arbitrary Fourier mapping and show that it improves generalization of the interpolation task. We weigh the frequencies of γ as follows:

$$\gamma^\alpha(\mathbf{x}) := \begin{pmatrix} w^\alpha_{\mathbf{B}} \\ w^\alpha_{\mathbf{B}} \end{pmatrix} \circ \gamma(\mathbf{x}) \tag{13}$$

where $w^\alpha_{\mathbf{B}}$ is the element wise application of the function $w_\alpha(z)$ on the vector of Norms of $\mathbf{B}$ on the input dimension:

$$w^\alpha_{\mathbf{B}} := w_\alpha \begin{pmatrix} ||B_1||_2 \\ \vdots \\ ||B_m||_2 \end{pmatrix}. \tag{14}$$

where $w_\alpha(z)$ is defined as:

$$w_\alpha(z) = \begin{cases} 0 & \text{if } \alpha - z < 0 \\ \frac{1-\cos((\alpha-z)\pi)}{2} & \text{if } 0 \leq \alpha - z \leq 1 \\ 1 & \text{if } \alpha - z > 1 \end{cases} \tag{15}$$

Here, $\alpha \in [0, \max((||B_i||_{d_{\text{in}}})_{i \in \{1,\ldots,m\}})]$ is a parameter which is linearly increased during training. This strategy forces the network to train the low frequencies at the start of training, ensuring that the network will produce smooth outputs. Later, when high-frequency activations are allowed, the low-frequency components are trained, and the network can focus on the left details. This strategy should reduce the effect of overfitting, which was introduced by Tancik et al. [32] when using mappings with large standard deviations.

3.4. Pruning

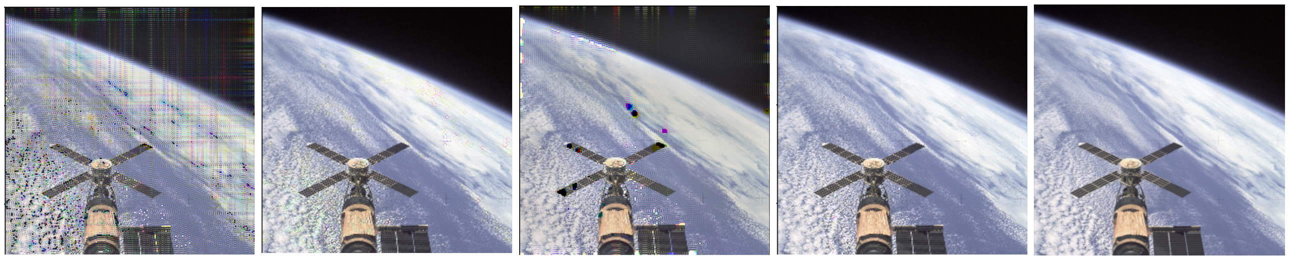
The standard way of using equation (10) is by defining a value N and taking the whole set $\mathbf{B}_N$. High-dimensional tasks lead to high memory consumption, and it is not clear whether this subset of $\mathbb{Z}^d$ brings best performance. We, therefore, propose a way to select a more appropriate subset through data pruning. A pruning $pr(N, M)$ is done as follows: Assume we have $N, M \in \mathbb{N}$ with $M >> N$ and $|\mathbf{B}_N| = n$, $|\mathbf{B}_M| = m$. We train a perceptron with an integer mapping given by $\mathbf{B}_M$. After training we define $\mathbf{D}$ such that $\mathbf{D}$ contains only those elements of $\mathbf{B}_M$ where the respective weights are greater than a margin, that is chosen to yield $|\mathbf{D}| = n$. While $\mathbf{B}_N$ and $\mathbf{D}$ now have the same size, we believe that $\mathbf{D}$ will yield better performance because it contains the most relevant frequencies of the signal we want to reconstruct.

3.5. Integer lattice mapping applied to MLPs

Although we showed in section 3.1 that we can represent any bounded input function with only one Fourier mapped perceptron, in practice, these networks can become very wide to give high performance. As a result, the number of calculations will increase. To compromise between performance and speed, one can add depth and reduce the width of the network.

First, it is natural that using MLPs rather than perceptrons increases the performance. However, it remains unclear why our proposed integer mapping should perform better than competing mappings for multilayer networks.

One could argue that if a mapping gives the perceptron a high representation power, it will also provide a high representation power to the MLP and vice versa. First, however, we should verify this claim with experiments. In addition, we remind the reader that a periodic function has integer frequencies. And because our assumption that the signal we want to reconstruct is periodic, it will have only integer frequencies. Also, the activation functions we are using only introduce integer frequencies when applied to a periodic function, as shown for the 1D case in the supplementary material. With this, we reduce the search space for frequencies from $\mathbb{R}$ to $\mathbb{Z}$, which could make the optimization easier as the search space is more compact and approachable.



(a) PT=F, WI=F (b) PT=F, WI=T (c) PT=T, WI=F (d) PT=T, WI=T (e) GT

Figure 2: A visualization of the outputs of Fourier mapped perceptrons of $N = 128$. PT stands for progressive training and WI stands for weight initialization. T/F stands for True/False, respectively.

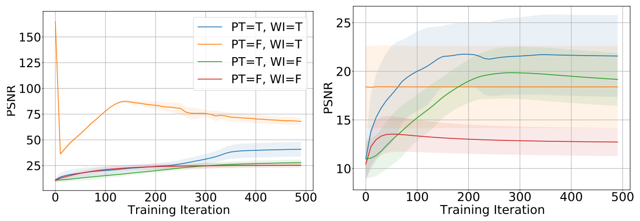


Figure 3: The training progress of Fourier mapped perceptrons with $N = 128$. The left and the right figures report the train and test PSNR, respectively. Weight initialization without PT yields a PSNR of 160 which one can consider as the ground truth proving that the perceptron is a Fourier series. Note: The y-axis limits are different in both plots.

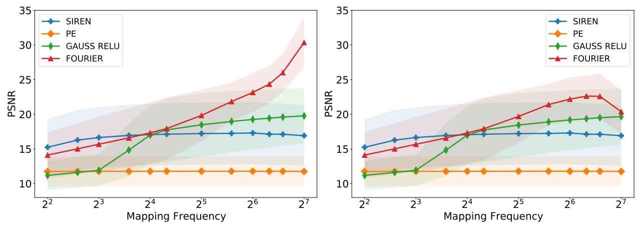


Figure 4: Perceptron experiments with different values for the mapping frequency N. We report the train PSNR on the left and the test PSNR on the right. For high values of N our integer mapping outperforms all competing mappings.

4. Experiments

4.1. Weight initialization and progressive training

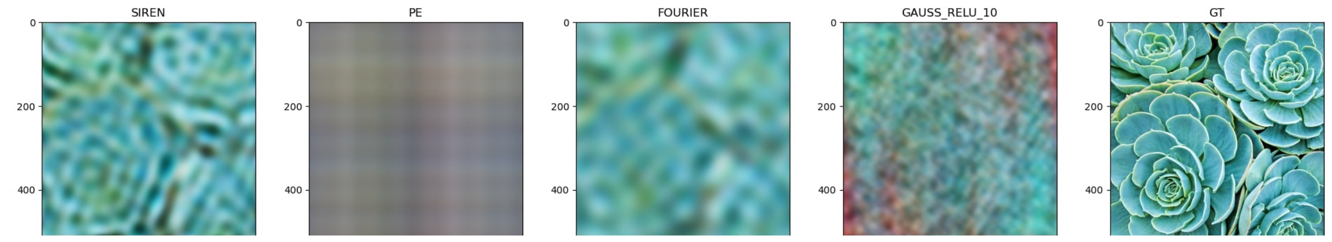
In this section, we want to prove our mathematical claims by experiments. First, we will show that the derivation of the integer mapping indeed represents the Fourier series. Secondly, we want to check whether progressive training helps with generalization.

We conducted our experiments on the image regression task. This task aims to make a neural network memorize an image by predicting the color at each pixel location. We use ten images with a resolution of 512×512, which can be found in the supplementary material, and report the mean peak signal-to-noise ratio (PSNR). We divide the image into train and test sets, where we use every second pixel for training and take the complete image for testing. We utilize 3 Fourier-mapped perceptrons with $N = 128$ (Nyquist frequency), one for each image channel. We normalize the input ($\mathbf{x}$) to have an interval between [0,1] in both width (x) and height (y) dimensions.

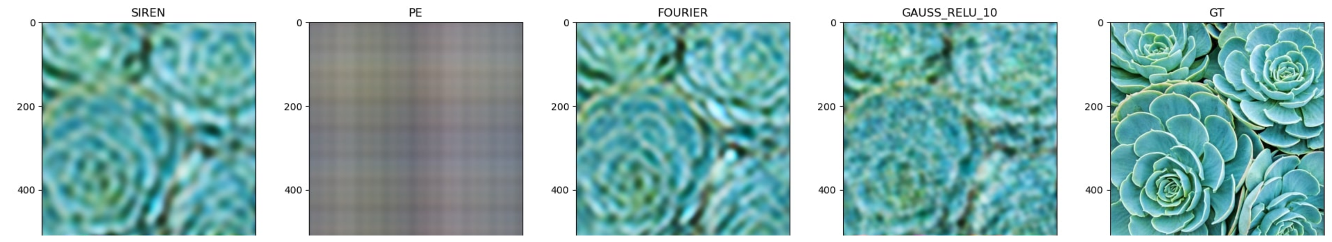
In this experiment, we made an ablation: With and without weight initialization using the normalized FFT coefficients of the image's training pixels, with and without the progressive training scheme explained in section 3.3. For progressive training, α was linearly increased from 0 to its maximum value at 75% of training iterations. In training, we only make an update step after we accumulate the gradients of the whole image. We did not study learning schedules in this work, and the reader is encouraged to try different schedules. Figure 2 shows a visualization of one of the images, and figure 3 shows the training progress, where the solid line is the mean PSNR and the shaded area shows the standard deviation.

As can be deducted from figure 3, one can see that the train PSNR starts at an optimum at the start of training when we use weight initialization (WI), and we don't use progressive training (PT). This fact underlines our claim that *a perceptron with an integer lattice mapping is indeed a Fourier series*. Note that in case both WI and PT are used, the train PSNR is not optimal at the start because the PT masks out high-frequency activations.

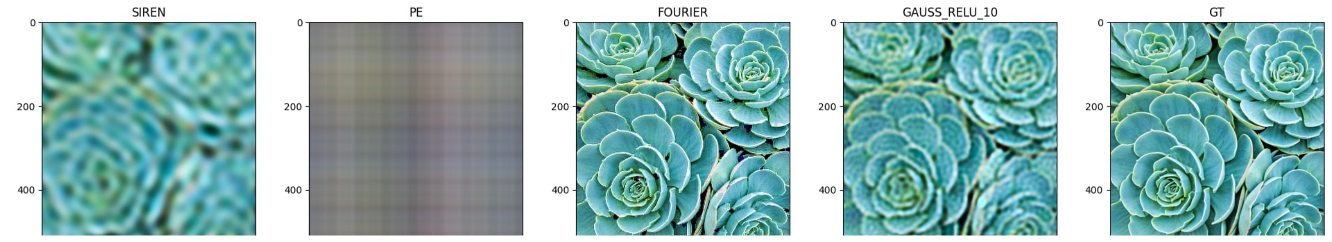
We can also see from figure 3 that whenever we use progressive training, it always shows a higher test PSNR, which certifies that *progressive training helps with generalization*. Lastly, when we did not employ both PT and WI, the perceptron overfits to the training pixels, and this can be seen quantitatively with a very low test PSNR (red line in figure



(a) Network predictions using N = 8



(b) Network predictions using N = 16



(c) Network predictions using N = 128

Figure 5: The visualization of the Fourier mapped perceptrons and the one layer SIREN with different values of N.

3) and qualitatively with grid-like artifacts (in figure 2a)).

4.2. Perceptron experiments

In this experiment, we want to compare the representation power of the different mappings in the single perceptron case. We conducted our experiments in the same setting as in section 4.1, where we used progressive training and did not use weight initialization.

In the integer mapping, we increased N's value from 4 until half the training image dimension (Nyquist frequency) and calculated all possible permutations $\mathbf{B}_N$, as discussed in section 3.1. For the Gaussian mapping, we sample $m = |\mathbf{B}_N|$ parameters from a Gaussian distribution with a standard deviation of 10 (which was the best value for this task in our experiments). Also, we test a one-layer SIREN with one hidden layer having the same size m. Finally, we adopt the positional encoding (PE) scheme from [19] and limit its values to N. Figure 4 shows our experiments' results on the train and test pixels, respectively. Figure 5 shows the networks' outputs trained on one of the images.

At low N values (figure 5a), we see that the Gaussian mapped perceptrons do not work because the number of sampled frequencies is low, so there is a low chance that samples will be near the image's critical frequencies. On the other hand, the integer mapped perceptrons give a blurry image because they can only learn low frequencies. The SIREN performs relatively well in this case, and we think this is because SIRENs naturally inherit a learnable Fourier mapping that is not restricted to the initial sampling, as described in section 3.2. PE can only produce horizontal and vertical lines because it has diagonal frequencies (only one non-zero frequency is allowed), and this effect is persistent at any value of N.

As N increases, SIREN, Gauss, and integer mapping performance increase giving similar performance around $N = 16$ (figure 5b). For high values of N, we see that in figure 5c, the integer lattice mapping of the Fourier coefficients outperforms the competing mappings, clearly displaying more details in the reconstruction. On the other hand, the PSNR of the SIREN and the Gaussian mapped perceptrons saturates. We think this is because both mappings rely on sampling the frequencies. Although we can get many of the critical frequencies of the image with sampling, it is improbable to get all of them simultaneously.

Activ.	Map.	N=8 m=113				N=16 m=481				N=32 m=1985			
		d=0	d=2	d=4	d=6	d=0	d=2	d=4	d=6	d=0	d=2	d=4	d=6
Sine	No	**16.65**	22.15	**23.26**	**24.07**	17.07	22.09	23.84	19.76	17.22	14.90	14.67	13.63
	Int.	15.68	**22.31**	22.41	20.94	**17.33**	27.66	27.06	**27.33**	**19.84**	33.78	26.98	23.60
	Pr.	15.28	21.03	22.40	23.00	16.76	**28.17**	**27.68**	24.66	18.48	37.34	30.41	19.74
Relu	P.E.	11.78	16.61	17.37	17.77	11.78	16.87	17.79	17.95	11.78	17.05	18.15	18.15
	Gs. σ_{10}	11.93	21.90	21.68	21.69	17.01	24.53	24.26	25.13	18.48	26.10	26.30	27.48
	Gs. σ_{pr}	14.06	20.23	20.78	20.88	12.69	26.02	26.40	26.72	13.01	37.69	**37.90**	**37.74**
	Int.	15.68	20.51	20.65	20.62	**17.33**	24.42	24.09	24.49	**19.84**	31.57	32.14	32.79
	Pr.	15.28	20.35	20.92	20.96	16.76	25.87	26.23	26.33	18.48	**37.70**	36.81	37.48

Table 1: The mean train PSNR results of network type comparison experiment with varying network depth (d), number of frequencies (N). We use the following abbreviations: Activ. = Activation function, Map. = Mapping type, Int. = Integer, Pr.= Pruned Integer, P.E. = Positional Encoding, Gs. = Gaussian. Here, m is the mapping size and σ is the standard deviation.

Activ.	Map.	N=8 m=113				N=16 m=481				N=32 m=1985			
		d=0	d=2	d=4	d=6	d=0	d=2	d=4	d=6	d=0	d=2	d=4	d=6
Sine	No	**16.65**	21.63	**21.85**	**21.99**	17.06	21.28	22.03	18.50	17.22	13.57	13.29	12.37
	Int.	15.68	**21.75**	21.53	20.06	**17.31**	**23.48**	**22.67**	22.28	**19.70**	16.85	17.89	16.36
	Pr.	15.28	20.49	21.22	21.45	16.75	22.00	21.39	22.17	18.39	20.49	15.15	13.13
Relu	P.E.	11.78	16.60	17.33	17.70	11.78	16.85	17.73	17.87	11.79	17.02	18.06	18.02
	Gs. σ_{10}	11.93	20.67	21.06	20.90	17.00	22.96	22.78	23.04	18.45	23.66	23.61	**23.73**
	Gs. σ_{pr}	14.06	19.89	20.22	20.21	12.69	22.46	22.48	22.16	12.99	23.12	23.48	23.33
	Int.	15.68	20.27	20.35	20.23	**17.31**	22.93	22.65	**22.50**	**19.70**	**24.36**	**24.02**	**23.73**
	Pr.	15.28	19.98	20.33	20.21	16.75	22.31	22.26	22.09	18.39	23.24	23.18	23.30

Table 2: The mean test PSNR results of network type comparison experiment. For abbreviations see table 1.

Even the trainability of the SIREN mapping did not help in this case.

4.3. MLP experiments

Our theory for integer mapping assumes an underlying function that is periodic. However, it is not clear that we will end up with a periodic function if we go the other way, using an integer mapping. In this experiment, we want to check if applying an integer mapping forces periodicity. Secondly, we want to validate our claim (in section 3.5) that if a mapping gives the perceptron a high representation power, it will also give a high representation power to the MLP and vice versa. We compared ReLU networks with integer, Gaussian, PE, and pruned integer mapping (section 3.4). We also compared SIRENS with no mapping (extra layer), integer, pruned mapping. We made a grid search of the parameters N=[8, 16, 32], depth=[0, 2, 4, 6] (depth=0 represents a perceptron), and fixed the width to 32. For the pruned mapping, we used a $pr(N, 128)$. And for the Gaussian mapping, we had two settings. The first one had a standard deviation of 10 (σ_{10}), which had the best performance in the perceptron experiments. In the second one, we set the standard deviation the same as the pruned integer mapping's standard deviation (σ_{pr}) to check its effect. Tables 1 and 2 show the mean train and test PSNRs respectively.

Figure 1 shows a visualization of the network's outputs at N = 16, depth = 4 and width = 32 for the first period and next period in the height and width directions ($f([x+1, y+1])$). And we see that *the integer mapping forces the network's underlying function to be periodic* unlike the SIREN and ReLU network with Gauss mapping, which proves our first hypothesis.

From the table 1 we see that if a mapping at $d = 0$ gives the highest PSNR, this does not mean that it will give the highest PSNR for $d > 0$ and vice versa. One clear example at $N = 32$ is the Gauss σ_{pr}, where it has a PSNR of 13.01 dB at $d = 0$, which is lower than integer mapping (19.84 dB), but has the highest PSNR at $d = [4, 6]$. This result disproves our initial assumption that if a mapping gives the perceptron a high representation power, it will also give a high representation power to the MLP. We see also that the pruned integer mapping has comparable results with the Gauss σ_{pr}, and this shows that the main contributor to the performance is the mappings' standard deviation.

From the tables, we can also observe some trends. First, networks with sine activations and large mappings collapse during perform worse than Relu networks. Second, the integer mapping usually gives the best test PSNR, demonstrating its effectiveness in the MLP case. Third, the pruned integer mapping shows consistently better train PSNR than

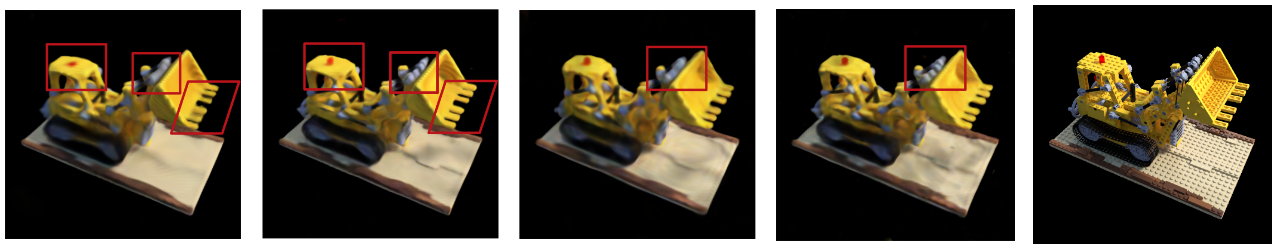

(a) ReLu without pruning. (b) ReLu with pruning. (c) Sine without pruning. (d) Sine with pruning. (e) GT

Figure 6: View synthesis results using a simplified Nerf. A small MLP with a depth of 4, width of 64 and integer mapping with a frequency of 4 is used. The pruning is done with pr(4,8). The pruning technique shows qualitative improvements.

Act.	Map.	$N = 4$				N=8
		d=0	d=2	d=4	d=6	d=0
Sine	No	**20.37**	23.08	**23.55**	23.35	OM
	Int.	18.42	22.22	22.95	22.97	**19.31**
	Pr.	19.15	**23.12**	**23.58**	23.36	-
Relu	P.E.	16.30	21.48	22.64	23.51	16.40
	Gs.	18.93	22.81	**23.64**	**23.82**	19.29
	Int.	18.42	21.81	22.68	23.28	**19.31**
	Pr.	19.15	22.78	23.61	**23.89**	**-**

Table 3: Validation PSNR scores of Nerf experiments using a mapping of frequency 4. OM stands for out of memory. For other abbreviations see table 1.

the normal integer mapping at $d > 0$. We believe this is because pruned mapping has a higher standard deviation. Finally, the PE is worse in every case because we cannot easily control the standard deviation, and it has very few parameters.

4.4. Novel view synthesis experiments

This section wants to see if our findings in the image regression task transfer to the novel view synthesis (NVS) task. In NVS, we are given a set of 2D images of a scene, and we find its 3D representation. With this representation, one can render images from new viewpoints. In contrast to the 2D experiments, the inputs are (x, y, z) coordinates that are mapped to a 4-dimensional output, the RGB-values, and a volume density. For this experiment, a simplified version of the official NeRF [19] is used, where the view dependency and hierarchical sampling are removed. Here, we experiment with the input mappings used in section 4.3. Unless otherwise stated, we adopt the settings from the image regression task. We set the network width to be 64.

As the mapping size increases exponentially, we do our experiments with lower frequencies than in the 2D case. We used the integer mapping on four frequencies. The frequencies of our mapping were limited to the maximum network size which we could fit on NVIDIA GTX-2080Ti. The pruning is given by $pr(4, 8)$. We conduct our experiments on the bulldozer scene, which is commonly used for Nerf experiments. For training, we used a batch size of 128, 50.000 epochs and a learning rate of 5×10^{-4}.

As seen in Table 3, in the perceptron case ($d = 0$), SIREN provides the best performance, which aligns with our image regression results at low values of N. We observe that the pruned mapping increases the performance compared to normal mapping for both Relu and sinusoidal activation. This increase in performance is because pruned mapping has a higher standard deviation than normal mapping. Qualitative improvements of the pruning can be seen in Figure 6. Gauss gives comparable results to pruned integer mapping because they have the same standard deviation. These findings align with our conclusions from image regression experiments. However, due to memory limitations, we could not test a perceptron with frequencies higher than 8, which was superior in image regression.

5. Conclusion

In this work, we identified a relationship between the Fourier mapping and the general d-dimensional Fourier series, which led to the *integer lattice mapping*. We also showed that this mapping forces periodicity of the neural network underlying function. From experiments, we showed that one perceptron with frequencies equal to the Nyquist rate of the signal is enough to reconstruct it. Furthermore, we showed that the progressive training strategy improves the generalization of the interpolation task. Lastly, we confirmed the previous findings that the main contributor to the mapping performance is its size and the standard deviation of its elements.

References

[1] Matan Atzmon and Yaron Lipman. Sal: Sign agnostic learning of shapes from raw data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2565–2574, 2020.

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

reconstruction. In *European Conference on Computer Vision*, pages 608–625. Springer, 2020.

[3] Zhiqin Chen. *IM-NET: Learning implicit fields for generative shape modeling*. PhD thesis, Applied Sciences: School of Computing Science, 2019.

[4] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019.

[5] Boyang Deng, John P Lewis, Timothy Jeruzalski, Gerard Pons-Moll, Geoffrey Hinton, Mohammad Norouzi, and Andrea Tagliasacchi. Nasa neural articulated shape approximation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16*, pages 612–628. Springer, 2020.

[6] A Ronald Gallant and Halbert White. There exists a neural network that does not make avoidable mistakes. In *ICNN*, pages 657–664, 1988.

[7] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local deep implicit functions for 3d shape. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4857–4866, 2020.

[8] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7154–7164, 2019.

[9] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *International Conference on Machine Learning*, pages 3789–3799. PMLR, 2020.

[10] Philipp Henzler, Niloy J Mitra, and Tobias Ritschel. Learning a neural 3d texture space from 2d exemplars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8356–8364, 2020.

[11] Arthur Jacot, Clément Hongler, and Franck Gabriel. Neural tangent kernel: Convergence and generalization in neural networks. In *NeurIPS*, 2018.

[12] 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.

[13] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. *arXiv preprint arXiv:2104.06405*, 2021.

[14] Shuang Liu. Fourier neural network for machine learning. In *2013 International Conference on Machine Learning and Cybernetics*, volume 1, pages 285–290. IEEE, 2013.

[15] Shichen Liu, Shunsuke Saito, Weikai Chen, and Hao Li. Learning to infer implicit surfaces without 3d supervision. *Advances in Neural Information Processing Systems*, 32:8295–8306, 2019.

[16] Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. Dist: Rendering deep implicit signed distance function with differentiable sphere tracing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2019–2028, 2020.

[17] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.

[18] Mateusz Michalkiewicz, Jhony K Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders Eriksson. Implicit surface representations as layers in neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4743–4752, 2019.

[19] 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. In *European Conference on Computer Vision*, pages 405–421. Springer, 2020.

[20] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *International Conference on Computer Vision*, Oct. 2019.

[21] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3504–3515, 2020.

[22] Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. Texture fields: Learning texture representations in function space. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4531–4540, 2019.

[23] Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. Texture fields: Learning texture representations in function space. In *International Conference on Computer Vision*, Oct. 2019.

[24] Brad G Osgood. *Lectures on the Fourier transform and its applications*, volume 33. American Mathematical Soc., 2019.

[25] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.

[26] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 523–540. Springer, 2020.

[27] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2304–2314, 2019.

[28] Adrian Silvescu. Fourier neural networks. In *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No. 99CH36339)*, volume 1, pages 488–491. IEEE, 1999.

[29] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33, 2020.

[30] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: continuous 3d-structure-aware neural scene representations. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 1121–1132, 2019.

[31] Kenneth O Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic programming and evolvable machines*, 8(2):131–162, 2007.

[32] 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. *Advances in Neural Information Processing Systems*, 2020.