Laboratório VISGRAF

Instituto de Matemática Pura e Aplicada

Sinusoidal Networks in practice:
Controlling the frequencies' factory

Diana Aldana, Tiago Novello, Luiz Velho

Technical Report TR-24-02 Relatório Técnico

January - 2024 - Janeiro

Sinusoidal Networks in practice:
Controlling the frequencies' factory

Diana Aldana, Tiago Novello, Luiz Velho

January 19, 2024

Abstract

Recently, (coordinate-based) *sinusoidal neural networks* have exhibited promising results in representing most signals in computer graphics, such as images and implicit surfaces. There is, however, a general lack of understanding behind the capacity and architecture a neural network may have to represent a determined signal. In this work, we study sinusoidal neural networks from a Fourier series perspective and link the initialization and training schemes with the model's *dominant frequencies* to obtain the appropriate capacity for the model to learn the signal.

Based on the observation above, we propose a *training procedure* that restricts the network's band limit during training. Additionally, we present a *pruning scheme* that explores the model's properties to compact its parameters during (or after) training. Finally, we propose a scheme that adapts the model size during training, allowing the network to accurately and compactly represent a signal.

1 Introduction

1.1 Motivation

Recently, there has been a growing surge in the use of coordinate-based neural networks –also known as implicit neural representations or INR– within the computer graphics community. In contrast to their classical discrete counterpart, where signals are sampled and vectorized before processing, INRs encode them continuously on the parameters. They do so by mapping the coordinates to the target signal values.

Tancik et al. [33], Yüce et al. [39] proved that by first mapping coordinates to a harmonic dictionary, coordinate-based neural networks bypass the *spectral bias* [22] that previously hindered the advance of representational *multilayer perceptron* neural networks (MLPs).

Specifically for the case of periodical neural networks, it was recognized to be a hard task to train such models [19] given their global support and infinite critical points. Sitzmann et al. [27] overcome this issue by defining a special initialization scheme that guarantees stability and convergence for *sinusoidal* neural networks, MLPs with *sine* activation function.

This result motivated several works [20, 21, 24] that empirically proved that these networks have a high capacity to represent fine details. This success is partially due to two principal properties of sinusoidal MLPs: smoothness and compactness.

However, determining/compacting the parameters of these networks remains an empirical task. This work considers an alternative approach to studying this problem through a novel dictionary learning approach that contemplates a pruning scheme with finetuning that guarantees training stability by targeting the most redundant weights and a grow model stage that adds Fourier atoms to the dictionary agreeing with the target signal's Fourier spectra.

Moreover, as seen in Tošić and Frossard [34], Yüce et al. [39], a recurrent problem during sinusoidal neural networks training has been the control of the reconstruction's frequency bandwidth. By imposing a condition over the hidden layer weights according to the upper bound shown in Novello [15], this work presents a method to control the growth of the model's bandwidth during training.

1.2 Contributions

This work relies on the theoretical background recently proposed by Novello [15], setting a mathematical foundation to address the problems of optimal model capacity, training stability, and fast convergence similarly to the ω_0 parameter. Finally, it proposes a compression scheme based on the insights obtained.

More specifically, our contributions are:

- Interpret a single hidden layer and the model's it's capacity (width and depth): The results shown in [15] allow us to understand the role of several model components on the reconstruction's Fourier series (see Chapter 4). Chapter 5 shows that it is possible to initialize the sinusoidal network to fit the signal with more precision and faster convergence while considering the significance of each component on the reconstruction.
- Frequency control: Based on the upper bounds defined in [15], we show in Chapter 6 that it is possible to grow the frequency band during training, allowing the model to progressively generate higher frequencies in line with the distribution of energy commonly seen in the spectrum of images.
- Compression: Using the understanding obtained on the model coupled with a custom initialization scheme, Chapter 7 explores an importance criterion for each weight in the model. With such criteria, we suggest a structured pruning scheme (see Definition 3.2.2) that will consider the relationship between weights in the same row(column) and their influence on the reconstruction.

2 Related Works

Implicit Neural Representations: Implicit Neural Representations (INRs) [36] are emerging topics of interest in the artificial intelligence community. In contrast to traditional representational methods, INRs act as a continuous mapping between coordinates and the signal values, codifying the target function into their parameters as fields, which makes them efficient and compact.

Current INR architectures use positional encoders, such as Fourier feature mappings [14, 33] or sinusoidal activation functions [27], to allow the model to bypass the spectral bias [22] common in MLPs, representing fine details. A more detailed study of the architecture's representational capacity was studied by Yüce et al. [39], Zell et al. [40], presenting the most common problems of INRs (appearance of artifacts or imperfect recovery), showing that initialization of the model increases encoding efficiency and modeling the fitting problem as a dictionary learning problem [34].

Periodic INR: Considering periodic activation functions in neural networks is an old approach [19, 30]. However, it wasn't until the introduction of the initialization schema proposed by Sitzmann et al. [27] that networks with periodic activation functions became widespread. They have shown promise in the representing multimedia such as audios [6, 27], images [2, 3, 5, 29], representation of signed distance functions [12, 16, 24, 27], displacement fields [38], surface animation [17], among others. In particular, being smooth representations allows us to use sinusoidal networks' analytical derivatives as constraints during the training (see [16]) or to represent solutions of differential equations [27].

Moreover, the novel approach presented in [15] defines an equivalent form to the model as a sum of Fourier atoms that showcase the role of the model components in the signal's Fourier series approximation. [40] presented a similar approach where it was shown the relationship between Fourier feature mapping, SIREN, and the Fourier series of the approximation when initializing the first layer with integer frequencies (as is the case in most of this work).

Control of frequencies: One of the most well-known downsides of sinusoidal networks is the noise generation during training or in networks with many hidden layers. Sitzmann et al. [27] bypassed the problem with an initialization scheme that kept the distribution of the weights from changing abruptly during training. However, the parameter ω_0, if not tuned well, introduced noise (high frequencies not present in the original function).

Along the same idea as SIREN surged WIRE [23], an INR that uses complex Gabor wavelet as an activation function given that it is optimally concentrated in space–frequency and has shown good results in classical image representation theory. Despite promising, this approach has yet to provide a definite advantage over sinusoidal networks.

BACON [12] presented a more straightforward strategy: A multiplicative network architecture where the Fourier spectrum characterizes the behavior of the trained model, and each layer's bandwidth can be adjusted analytically. Recently, Dou et al. [4] added a level of detail scheme to reduce noise by centering high frequencies in sections of the domain where the signal had the most details.

Compression on neural networks: Previous to the boom of coordinate-based neural networks, different works already focused on model compression. Several methods [5, 28] consist of a mixture of pruning, quantization, and entropy coding scheme to reduce the model, while others use knowledge distillation [32], or meta-learning to create a residual representation of the model [25, 31]. Menghani [13] gives a more extensive survey of deep learning compression schemes.

3 Preliminaries

This work addresses the problem of obtaining an **efficient INR** given any sampled signal. In particular, we focus primarily on image media, narrowing the study to signals $f : \mathbb{R}^2 \to \mathbb{R}^3$. However, our methods extrapolate to other domains and codomains.

We will describe the INR used throughout this work and several implementation choices. Then, we define what we shall consider an efficient INR and depict the methods employed to fit our model with that objective in mind.

3.1 Representation

We focus on the subclass of INRs that consists of *multilayer perceptrons* (MLPs) that use the *sine* as activation function – *sinusoidal INRs* [27]. Such networks are compact compositions of multiple sinusoidal layers with high representational capacity.

Definition 3.1 (Sinusoidal Layer)**.** *Given $\boldsymbol{W} \in \mathbb{R}^{n\times m}$ and $\boldsymbol{b} \in \mathbb{R}^n$, a sinusoidal layer is a function $\boldsymbol{S} : \mathbb{R}^m \to \mathbb{R}^n$ defined as $\boldsymbol{S}(\boldsymbol{x}) = \sin(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$. The matrix $\boldsymbol{W}$ will be called weights, and $\boldsymbol{b}$ is the bias.*

Then, given a set of parameters $\theta = \{\omega := \mathbf{W}_0, \varphi := \mathbf{b}_0, \mathbf{W}_1, \mathbf{b}_1, ..., \mathbf{W}_l, \mathbf{b}_l\}$ such that $\mathbf{W}_i \in \mathbb{R}^{n_i\times n_{i-1}}$ and $\mathbf{b}_i \in \mathbb{R}^{n_i}$ ($n_0 = 2$, $n_l = 3$, $n_i \in \mathbb{N}^+$ for $0 < i < l$), we can define a sinusoidal neural network as:

$$f_\theta(\mathbf{x}) = \mathbf{W}_l\mathbf{S}_{l-1} \circ \cdots \circ \mathbf{S}_1 \circ \mathbf{S}_{\omega,\varphi}(\mathbf{x}) + \mathbf{b}_l \tag{1}$$

where l is the *depth* of the model and n_i is denoted the *width* of the (sinusoidal) layer i.

The first layer, which will be of particular importance, will be specially denoted as $\mathbf{H} = \mathbf{S}_{\omega,\varphi}$. Layer l is usually noted as *last/final* layer, and all others receive the name of *hidden layers.*

Notice that by introducing biases in each sinusoidal layer, it becomes equivalent to use *sine* or *cosine* as the activation function, given that

$$\sin(\mathbf{W}\mathbf{x} + \mathbf{b}) = \cos\left(\mathbf{W}\mathbf{x} + \mathbf{b} - \frac{\pi}{2}\mathbf{1}\right),$$

then a network with a sine activation function is equivalent module shifting of all biases to another model with a cosine activation function.

Understanding how sinusoidal layers perform is a recent research topic [15, 39, 40]. First, we note that $\mathbf{H}$ maps the coordinates $\mathbf{x}$ to a list of harmonics with frequencies defined by the coordinates of ω. Composing such a list with a hidden sinusoidal layer results in iterative harmonic expansions, also in terms of ω (Theorem 1). Finally, the linear transformation $\mathbf{W}_l \cdot \mathbf{x} + \mathbf{b}_l$ maps back to the original spatial space.

The above observation regarding frequency expansion implies that the coefficients of the first layer, namely ω and φ, are crucial. We refer to them as the *input frequencies* and *shifts*, respectively. Additionally, note that the *initialization* values for these coefficients define the type of positional encoding [27, 33] used by the model. Moreover, the initialization of θ parameters significantly influences training speed [8] and reconstruction quality.

This work proposes the following initialization settings:

- Based on Nyquist limit [18, 26], we select the initial harmonic *atoms* $\mathbf{H}_i$ ($i \in \{1, ..., n_0\}$) to have frequencies on the bandwidth $[-w_0, w_0]^2$ with $w_0 \in \mathbb{N}^+$. This new hyperparameter will have a similar role in the training as the ω_0 defined by Sitzmann et al. [27].
- The input frequencies are selected such that

  $$\omega \in \left(\frac{2\pi}{p}\mathbb{Z}^2 - \{(0,0)\}\right)^{n_0}.$$

  This guarantees (see Chapter 4) that the reconstruction will belong to the space of periodic functions with period p. This hypothesis, used by several contemporary works [11, 35, 37], is reasonable for image representation as the values of the function outside the image domain are irrelevant in the reconstruction task. Moreover, such models may aid in tasks like texture tiling or extrapolating the image's features.
- The input frequencies ω aren't updated during the training process, which guarantees the periodicity of the trained neural network (see

Theorem 1). Consequently, the reconstruction space is determined implicitly through the first layer initialization.

- Most results mentioned in this work can be inductively applied to a model with arbitrary depth $l > 1$. Therefore, all discussions will consider sinusoidal neural networks with a single hidden layer unless otherwise stated.

  The notation used henceforth will be $f_\theta(\mathbf{x}) = \mathbf{C}\mathbf{S}_{\mathbf{A},\mathbf{b}} \circ \mathbf{H}(\mathbf{x}) + \mathbf{d}$

Chapter 4 will give a more comprehensive overview of initialization and the model's frequencies.

3.2 Efficient Representations

It shall be considered as efficient a representation that is fast to compute, of high fidelity (usually measured in PSNR), and consumes less memory footprint. In that sense, sinusoidal networks have shown potential as efficient representations for several media. Moreover, with a careful selection of hyperparameters, the model overfit signals faster than most INRs.

Following the previous definition of efficient sinusoidal networks, the following sections will cover the descriptions of the training scheme and loss functionals deployed. Then, the compression method used in the experiments in Chapters 7 & 8 will be introduced.

3.2.1 Training & losses

When fitting an image, an iterative optimization process commonly known as *training* is applied over the sinusoidal neural network's parameters θ. In the n-th training step (or *epoch*), the function f_{θ_n}described in Equation 1 is compared against the true image on the known samples by using a *loss functional* $\mathcal{L}$. As the objective is to reduce the difference (loss) between f and f_{θ_n}, the parameters in θ_n are adjusted by $\Delta\theta_n$ using a gradient descent scheme over $\mathcal{L}$ (that is dependant on θ_n), resulting in the adjusted model $f_{\theta_{n+1}} = f_{\theta_n + \Delta\theta_n}$.

In this work, we use the classic Mean Square Error (MSE) loss function $\mathcal{L}_{MSE}$ to fit the model to the image. In addition, we employ the loss functions defined below during the compression process.

Definition 3.2 (Weight decay)**.** *Let f_θ be a sinusoidal neural network with depth 2. The weight decay loss functional is,*

$$\mathcal{L}_{wd}(f, f_\theta) = \|\boldsymbol{A}\|_1 .$$

The training process with loss functional $\mathcal{L} = (1-\alpha)\mathcal{L}_{MSE} + \alpha\mathcal{L}_{wd}$ with $\alpha \in [0,1]$ is denoted weight decay.

Definition 3.3 (Targeted weight decay)**.** *Let f_θ be a sinusoidal neural network with depth 2. Given $\mathcal{I} \subseteq \{1, ..., n_0\}$, the targeted weight decay loss functional (over columns) is defined as*

$$\mathcal{L}^c_{\mathcal{I}}(f, f_\theta) = \sum_{i \in \mathcal{I}} \left\| \boldsymbol{A}^{(i)} \right\|_1$$

where $\boldsymbol{A}^{(i)}$ refers to the i-th column of the matrix $\boldsymbol{A}$. The training process with loss functional $\mathcal{L} = (1-\alpha)\mathcal{L}_{MSE} + \alpha\mathcal{L}^c_{\mathcal{I}}$ with $\alpha \in [0,1]$ is called targeted weight decay in columns over the set $\mathcal{I}$.

Observation 1. *If $\mathcal{I} = \{1, ..., n_0\}$ then $\mathcal{L}^c_{\mathcal{I}} = \mathcal{L}_{wd}$.*

Observation 2. *By exchanging the columns of $\boldsymbol{A}$ for its rows in Definition 3.3, we obtain an equivalent definition for the targeted weight decay loss functional over rows, $\mathcal{L}^r_{\mathcal{I}}$.*

The previously defined loss functionals will have relevance when defining the pruning scheme in Chapter 7.

3.2.2 Compression methods

There are three popular techniques for neural compression: pruning, quantization, and distillation (see [10, Lectures 3-9]). This work will focus on a pruning scheme, a method classified as a lossy, data-independent compression method. However, future works may study other schemes like quantization, distillation, or lossless methods to obtain a more compact yet high-quality representation.

To be more precise, pruning refers to a method that assigns the value zero to the parameters in θ that meet specific conditions. In the context of this research, we will focus on the following definitions for pruning.

Definition 3.4 (Structured pruning)**.** *Given a sinusoidal neural network f_θ, and $\boldsymbol{A} \in \mathbb{R}^{n \times m}$ the middle weights.*

1. *Under a row pruning criteria $PC^r : \mathcal{M}_{n\times m}(\mathbb{R}) \to \{0,1\}^n$, the pruning of rows is defined as the update of $\boldsymbol{A}$ for $\boldsymbol{A} \cdot diag(PC^r(\boldsymbol{A}))$.*

2. *Given a column pruning criteria $PC^c : \mathcal{M}_{n\times m}(\mathbb{R}) \to \{0,1\}^m$, the pruning of columns is defined as the update of $\boldsymbol{A}$ for $diag(PC^c(\boldsymbol{A})) \cdot \boldsymbol{A}$.*

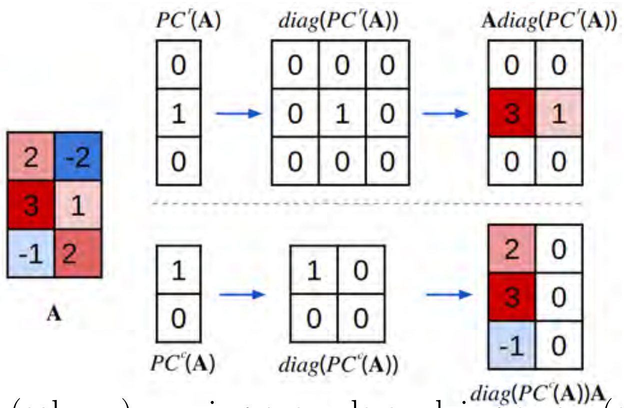

Figure 1: Row (column) pruning example applying a row (column) criteria over the weights $\mathbf{A}$.

Figure 1 shows an example of row and column pruning over the same parameters θ. As the zero-ed rows (columns) of $\mathbf{A}$ hold no information, they and all other weights canceled as a consequence of their change can ignored during the storing process. For example, in the case of column pruning shown in Figure 1, we could store a 3×1 array for $\mathbf{A}$ instead of the original 3×2 matrix; besides, the expression $\mathbf{H}_2$ is unused as it multiplies with the null column. Then, we could also ignore the second row of ω and φ.

Several criteria for pruning a weight may depend on the task at hand. We consider the weights *information* as their L_1 norm (see Figure 2), and the *pruning criteria* evaluated on the weights as the vector with:

- Zeros in the indexes of the rows (columns) with the least information such that their added information constitutes less or equal to $P\%$ of the total information.

- Ones on the remaining entries.

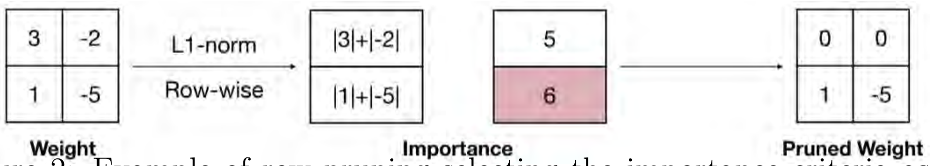

Figure 2: Example of row pruning selecting the importance criteria as the L1 norm of $\mathbf{A}$. Taken from the Han et al. [10, Lecture 3].

4 Interpretation

The previous chapter defined the concepts and mechanisms used in this work to create an efficient INR. This chapter provides an overview of the key findings from Novello [15] applied in this work. It also exposes the insights derived from those results to reinforce the results obtained from our methods. The proofs for all theorems enunciated in this chapter are found on [15].

Precisely, we show that a sinusoidal neural network closely resembles a Fourier series with coefficients, amplitudes, and phases derived analytically from the components of the neural network. That connection helps understand the relationship between input frequencies and *generated frequencies*. Then, the upper bound defined in [15] serves as an inspiration to establish the training schedule presented in Chapter 6.

For the following theorem let's consider $\widetilde{\mathbf{x}} = [\mathbf{x}, 1]^T$ and $\tilde{\mathbf{A}}, \Omega$ the augmented matrices of $\mathbf{A}$ and ω, respectively.

Theorem 1. *The coordinates* $h_i(\boldsymbol{x}) = \sin\big(\tilde{\boldsymbol{A}}\sin(\Omega\widetilde{\boldsymbol{x}})\big)$ $(\tilde{\boldsymbol{A}} \in \mathbb{R}^{n\times(m+1)})$ *of the sinusoidal MLP* f_θ*'s hidden layer can be rewritten as*

$$h_i(\boldsymbol{x}) = \sum_{\boldsymbol{k}\in\mathbb{Z}^n} \alpha_{\boldsymbol{k}}(\boldsymbol{A}_j)\sin\big(\langle\boldsymbol{k}, \Omega\widetilde{\boldsymbol{x}}\rangle + b_j\big) \tag{2}$$

where $\alpha_{\boldsymbol{k}}(\boldsymbol{A}_j) = \prod_{i=1}^{m} J_{k_i}(A_{ji})$ *is the product of the Bessel functions of the first kind evaluated over the elements of the row* $\boldsymbol{A}_j$*.*

By separating the bias term inside the sine function, it's possible to obtain a Fourier series-like representation,

$$f_\theta^i(\mathbf{x}) = \mathbf{C}_i\left[\sum_{\mathbf{k}\in\mathbb{Z}^n}\left(\widehat{A}_{\mathbf{k}j}\sin\big(\beta_{\mathbf{k}}(\omega)\mathbf{x}\big) + \widehat{B}_{\mathbf{k}j}\cos\big(\beta_{\mathbf{k}}(\omega)\mathbf{x}\big)\right)\right]_{j=1}^{n} + d_i \tag{3}$$

where $i \in \{1,2,3\}$, $\beta_{\mathbf{k}}(\omega) = \langle\mathbf{k}, \omega\rangle$ and $\widehat{A}_{\mathbf{k}j} = \alpha_{\mathbf{k}}(\mathbf{A}_j)\cos(\langle\mathbf{k}, \varphi\rangle + \mathbf{b})$, $\widehat{B}_{\mathbf{k}j} = \alpha_{\mathbf{k}}(\mathbf{A}_j)\sin(\langle\mathbf{k}, \varphi\rangle + \mathbf{b})$ are the coefficients associated with sine and cosine, respectively. An important fact to highlight is that Equation 3 is not the neural network's Fourier series as each *generated frequency* $\mathfrak{f}$ can be obtained by different linear combinations $\langle \mathbf{k}, \omega \rangle$ of the input frequencies.

Therefore, the coefficients $\widehat{A}_{\mathbf{k}j}, \widehat{B}_{\mathbf{k}j}$ are not the Fourier coefficients of f_θ. It would be necessary to find all solutions for the diophantine equation $\mathbf{X}\omega = \mathfrak{f}$ for each $\mathfrak{f} \in \mathbb{N}$ to obtain the Fourier representation for f_θ by aggregating all coefficients associated with the same frequency,

To analyze the implications of Theorem 1, the following toy example will show a small neural network trying to fit a simple signal with and without a-priori knowledge obtained from Equation 2.

Example 4.1. *Given a periodic signal $f : \mathbb{R} \to \mathbb{R} : x \mapsto \sin(\frac{2\pi}{p}x)$ with period p, and a sinusoidal neural network $f_\theta(x) = c\sin(a\sin(\omega x + \varphi) + b) + b$, fit the neural network such that $f \approx f_\theta$.*

By using Equation 2, we would obtain

$$f_\theta(x) = c\sum_{k\in\mathbb{Z}} \alpha_k(a)\sin\left(k\omega\frac{2\pi}{p}x + \big(k\varphi + b\big)\right) + d$$

As a result, employing the Fourier theory simplifies the problem by equating all frequencies' sine (and cosine) coefficients of both functions. But notice that ω is a fundamental factor in deciding the frequencies appearing on f_θ. Therefore, $\omega = \frac{2\pi}{p}$ is a prudent initialization.

- ***k=0:*** *By using the Fourier transforms we obtain:*

$$\hat{f}_\theta(0) = c\alpha_0(a)\sin(b) + d = 0 = \hat{f}(0)$$

and thus

$$d = -c\alpha_0(a)sin(b) \tag{4}$$

- ***|k|=1:*** *Based on Equation 2 follows the equality below,*

$$c\alpha_1(a)\boldsymbol{S}_{\omega,\varphi+b}(x) + c\alpha_{-1}(a)\boldsymbol{S}_{-\omega,-\varphi+b}(x) = \sin\left(\frac{2\pi}{p}x\right)$$

and using the facts that $\omega = \frac{2\pi}{p}$, $\alpha_{-1}(a) = J_{-1}(a) = -J_1(a) = -\alpha_1(a)$, sine is an odd function along with the sum of angles for sine, the formula above becomes,

$$2c\alpha_1(a)\sin\left(\frac{2\pi}{p}x + \varphi\right)\cos(b) = \sin\left(\frac{2\pi}{p}x\right)$$

which implies the following equations,

$$2c\alpha_1(a)\cos(b) = 1 \tag{5}$$

$$\varphi = 2l \quad for \quad l \in \mathbb{Z} \tag{6}$$

- ***|k|=2m:*** *Repeating the process for the case $|k| = 1$ and considering that $J_{-2m}(a) = J_{2m}(a)$ the result would be,*

$$2c\alpha_{2m}(a)\cos(b)\sin\left(2m\frac{2\pi}{p}x + 2m\varphi\right) = 0$$

so,

$$2c\alpha_{2m}(a)\cos(b) = 0 \tag{7}$$

- ***|k|=2m+1:*** *By considering that $J_{-(2m+1)}(a) = -J_{2m+1}(a)$, the equality would be,*

$$2c\alpha_{2m+1}(a)\sin(b)\cos\left((2m+1)\frac{2\pi}{p}x + (2m+1)\varphi\right) = 0$$

and thus,

$$2c\alpha_{2m+1}(a)\sin(b) = 0 \tag{8}$$

From Equality 5 it's clear that $c \neq 0$ and $\cos(b) \neq 0$ and that implicates that in Equality 8 $\alpha_{2m+1}(a) = 0$. From the infinite possibilities for φ and b, we select $\varphi = 0 = b$ as the nearest value (with high probability) from their initializations that satisfies Equalities 6 and 7 (for all $m \in \mathbb{Z}$), and that choice forces $d = 0$ in Equality 4. Finally, from Equality 5 we conclude that $c = 1/(2\alpha_1(a))$.

However, the value assigned for a is undecided as finding the same root for several first order Bessel functions is an open problem until now. Thus, the neural network will be an approximation for the target signal. Despite that, the behavior of first order Bessel functions near the origin indicates that a good approximation for a would be close to the origin, where $J_k(x) > J_l(x)$ if $k < l$ ($|x| < 1$ and $x \neq 0$) and all functions decay exponentially. Nevertheless, the initialization of c is close despite $c = 1/(2\alpha_1(a))$ for $a << 1$ (meaning $\alpha_1(a) << 1$) which slows down training as the update Δc is small at each epoch. By initializing $c = 1/a$ from the beginning, Figure 3 shows faster convergence with notably better quality.

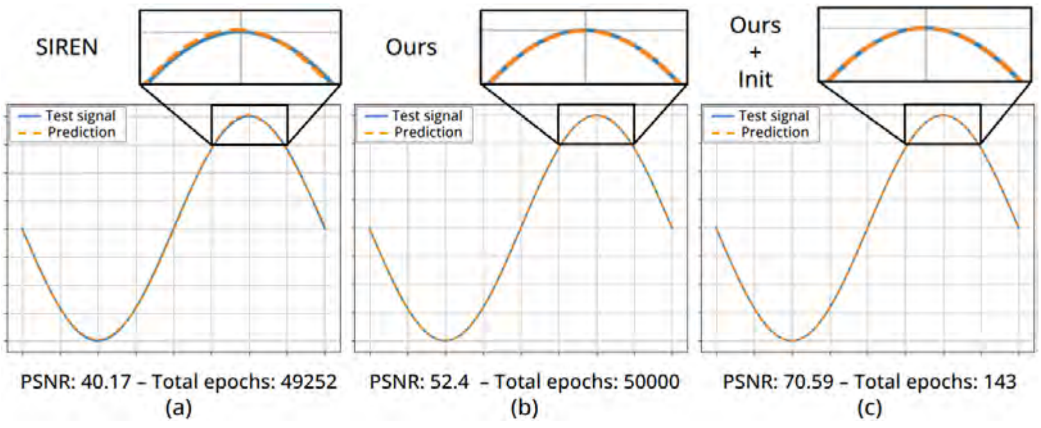

Figure 3: Fitting of target function $f(x) = \sin(\frac{2\pi}{p}x)$ with $p = 2$ during 50000 epochs using different initializations for f_θ. A premature stop indicates a reduction in the loss functional less than $1e^{-9}$. (a) SIREN initialization with fine-tuned $\omega_0 = 10$ has the worst performance. (b) Initialization with $\omega = \frac{2\pi}{2}$ has a good performance. (c) Model initialized with all considerations in the example and training only parameters a and c demonstrates a much better quality (+18 PSNR) with few training steps.

Despite being an over-simplified, well-behaved case, it's possible to extrapolate several ideas from the previous example to obtain a more efficient representation.

First, we chose ω based on the (known) frequency present in the signal. In this particular example, if $\omega \neq \pm\frac{2\pi}{p}$, a good representation would have been impossible as the frequency one would not have been present on the series defined by Equation 2. When f_θ where $m > 1$, this doesn't necessarily happen as input frequencies may introduce generated frequencies. However, initializing with the target signal's *dominant frequencies* guarantees smoother training (see Theorem 2).

Rather than having trouble fitting the target signal frequencies, the attenuation of frequencies not present (*noise*) in the original signal seems to pose a bigger problem, as seen in the previous example. Indeed, by considering only frequencies $\langle \mathbf{k}, \omega \rangle$ such that $|\mathbf{k}|_\infty < B$, Novello [15] deduced that f_θ may consider $\frac{(2B+1)^n - 1}{2}$, an exponential number of (potentially different) generated frequencies.

The generation of unwanted frequencies presents the necessity to control the network's bandwidth, centering the representation's energy along low frequencies. An implicit mechanism used in the example to handle this was to consider a small enough value for a coupled with a correct initialization of input frequencies, as third and higher order Bessel functions are practically zero around a small neighborhood around the origin. This indirect observation will become a key fact in the bound scheme presented in Chapter 6.

In a more general sense, it is possible to use Theorem 1 to get a rough inkling of the INR's anatomy and the role of each component during training. It's evident from Equation 2 that generated frequencies are determined by the input frequencies via linear combination, with amplitudes dependant on the rows A_j as well as first and hidden biases $\mathbf{b}$ and φ (see Novello [15, Corollary 1]). Adding the *shift* φ on the initial sinusoidal layer transforms it into a vector of harmonics with frequencies and phases determined by ω and φ, respectively. Finally, Equation 2 shows that the final matrix $\mathbf{C}$ weights the terms of the n infinite sums and $\mathbf{d}$ adjusts the network's mean.

The closed formulation of coefficients $\alpha_{\mathbf{k}}(\mathbf{A}_j)$ is used in Novello [15] to determine an upper bound, which shows that if $|\mathbf{A}_j|_\infty$ is small, the coefficients $\widehat{A}_{\mathbf{k}}$ and $\widehat{B}_{\mathbf{k}}$ decay exponentially fast as $|k|_\infty$ grows.

Theorem 2. *The amplitude $\alpha_{\boldsymbol{k}}(\boldsymbol{a})$ associated with frequency $\langle \boldsymbol{k}, \omega \rangle$ in Equation 2 satisfies the following inequality:*

$$|\alpha_{\boldsymbol{k}}(\boldsymbol{a})| < \prod_{i=1}^{m} \frac{\left(\frac{|a_i|}{2}\right)^{|k_i|}}{|k_i|!}$$

Consider the *order* of a frequency $\langle \mathbf{k}, \omega \rangle$ as the integer $|\mathbf{k}|_\infty$ and $|\mathbf{A}|_\infty < 1$. Then Theorem 2 would indicate that

$$|\alpha_{\mathbf{k}}(\mathbf{A}_j)| < \prod_{i=1}^{m} \frac{1}{2^{|k_i|}|k_i|!} \tag{9}$$

for all $\mathbf{k} \in \mathbb{Z}^m$, showing that higher order frequencies have negligible coefficients. Therefore, most of the energy is concentrated around low order frequencies, rendering the selection of input frequencies fundamental for the fitting task. Yang et al. [37] described this phenomenon, refering to it as the *spectrum concentration property* of INRs.

The initialization proposed Sitzmann et al. [27] that made it possible the use of sinusoidal neural networks in practice follows both principles stated before, initializing $\mathbf{A} \sim \mathcal{U}\left(-\sqrt{\frac{6}{m}}, \sqrt{\frac{6}{m}}\right)$ and $\omega \sim \omega_0 \mathcal{U}([-1/2, 1/2])$ with ω_0 a tunable hyperparameter. The first initialization satisfies the conditions for Equation 9 (as most applications require $m >> 1$), while in the second one, ω_0 scales the (real-valued) frequencies to an appropriate range.

The observation regarding high order frequencies might prompt us to initialize the input frequencies with low frequencies. By doing this, the model will prioritize fitting low frequencies first and gradually increase the influence of higher order frequencies during training aligned with spectral bias' theory. A deeper discussion about how to choose the initial values for the parameters θ will be held in the next section.

5 Initialization

Section 3.1 presented an initialization scheme aligned to the theory developed by Zell et al. [40], where the first sinusoidal layer is a vector of harmonics with fixed integer frequencies. Based on Theorem 1, ω determines the frequencies that could appear in the reconstruction's spectrum of a model with such a setting.

A naive approach to establishing the initial frequencies may consider computing the image's FFT and initializing ω with the dominant frequencies of the signal. However, the success of models such as SIREN shows that sinusoidal neural networks have the potential to learn the signal's spectrum with no other information about the target.

The next chapter will address the influence of input frequencies' choice and the model's capacity to approximate the target spectrum with a restricted bandwidth. Then, Section 5.3 will present problematic cases that may arise from inadequate initialization. Finally, Section 5.2 discusses the impact of ω over the fitting.

5.1 Frequencies' Generation

This section will address the model's prowess in learning new frequencies based on the initialized ones. Figure 4 shows that a sinusoidal neural network is capable of fitting a signal with high frequencies (i.e., $f(x) = \sum_{k=1}^{100} \sin(k\pi x)$) despite all initial harmonic atoms having frequency one.

A more intuitive demonstration of the increase of frequencies introduced by each hidden layer is given in Figure 4. There, two models with the same amount of parameters but with one and two hidden layers learn the same signal. In the first stage of training, the deeper model's FFT prediction presents a broader bandwidth than its counterpart, being able to introduce higher order frequencies.

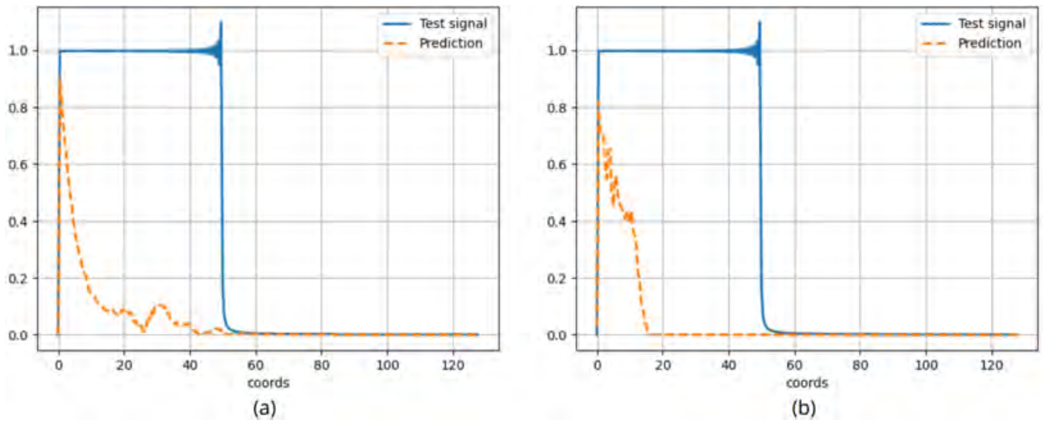

Figure 4: FFT prediction of a neural field trained to fit the signal $\sum_{k=1}^{100}\sin(k\pi x)$ during 500 epochs. Initialization of input frequencies $\omega = \mathbf{1}$. (a) Prediction of a sinusoidal network with depth 3 and 2995 parameters. (b) Prediction of a single hidden layer's sinusoidal network with 2996 parameters.

The previous analysis agrees with the common practice of favoring deeper architectures instead of wider ones and also exhibits commonly known problems of sinusoidal neural networks: Deeper models are prone to noisy reconstructions and are unstable to train.

A factor that may influence those phenomena is the lack of mechanisms to control the generation of high-order frequencies when the model is overparametrized. We consider two approaches to tackle this problem: Defining an adept initialization and employing a training scheme. Chapter 6 will present a methodology to tackle this problem using a training scheme, and the following section will analyze the initialization as a tool to avoid a noisy representation.

5.2 Initialization Experiments

It's natural to consider custom initial weights for each signal that accounts for its bandwidth and frequency content. However, finding such initialization may be costly and doesn't escalate well with higher dimension signals.

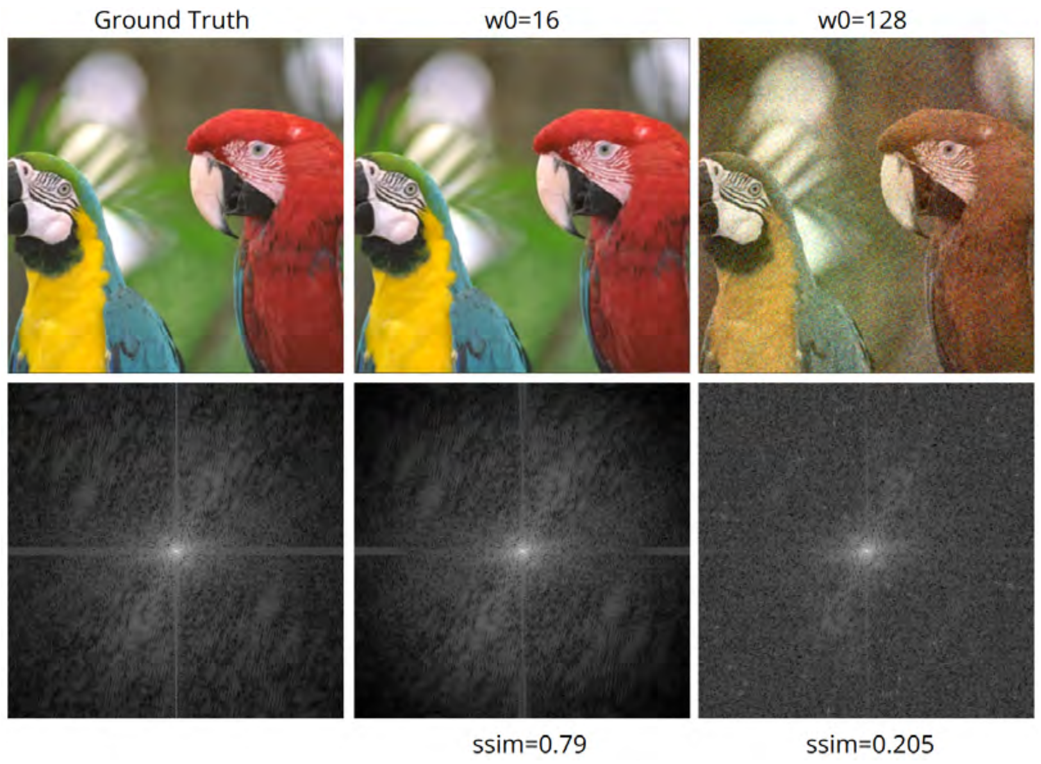

Figure 5: Reconstruction of networks with different bandwidths. The lower row presents the FFTs computed from the images above. (Left) Rasterized image used as training data. (Middle) Model with input frequencies randomly selected from the interval $([-16, 16] \cap \mathbb{Z})^2 - (0, 0)$. (Left) Model with integer input frequencies randomly chosen over the Nyquist Limit (128).

For instance, Figure 5 demonstrates that the straightforward strategy of randomly selecting input frequencies with $\omega_0 = 128$ (the Nyquist limit) leads to a noisy reconstruction. Since the input frequencies are chosen uniformly up to the maximum frequency on the image, and the network generates multiples of these input frequencies, frequencies beyond the Nyquist limit are likely to appear in the reconstruction.

On the other hand, selecting an adequate value for ω_0 emerges as a mechanism to control the generated Fourier spectrum (see Figure 5, middle's FFT prediction), leading to a better quality reconstruction.

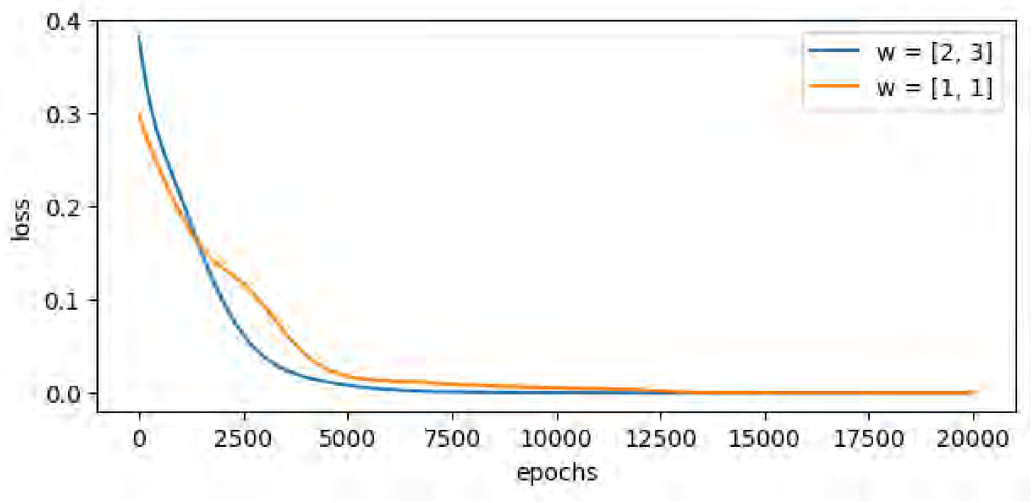

Figure 6: Loss evolution of single hidden layer's neural networks with $\omega \in \mathbb{R}^2, \mathbf{A} \in \mathbb{R}^{2\times 2}$ fitting the signal $f(x) = 0.5\sin(2\pi x) + 0.5\cos(3\pi x)$.

When information about the signal spectrum is available, it becomes possible to define ω by picking its $n-$th dominant frequencies. Figure 6 shows that the model converges faster with such coefficients, but they aren't indispensable for a good reconstruction.

Considering the spectral bias and the sinusoidal neural networks' ability to generate exponentially new frequencies, this work uses a small bandwidth $\omega_0 \leq 30$ for its experiments unless otherwise stated.

5.3 Problematic Cases

Although we have been selecting the input frequencies with no restrictions, determined initializations may hamper training or render reconstruction impossible. We present two problems that may arise from inadequate initialization and ways to detect and solve such issues.

First, observe that Figure 7(b) shows a reconstruction given by a sinusoidal neural network of period two that presents a problem with the reconstruction's period. Indeed, observing the extrapolation of the image (see Figure 7(c)), a repetition pattern is discerned along the diagonals (defined by white, dashed lines).

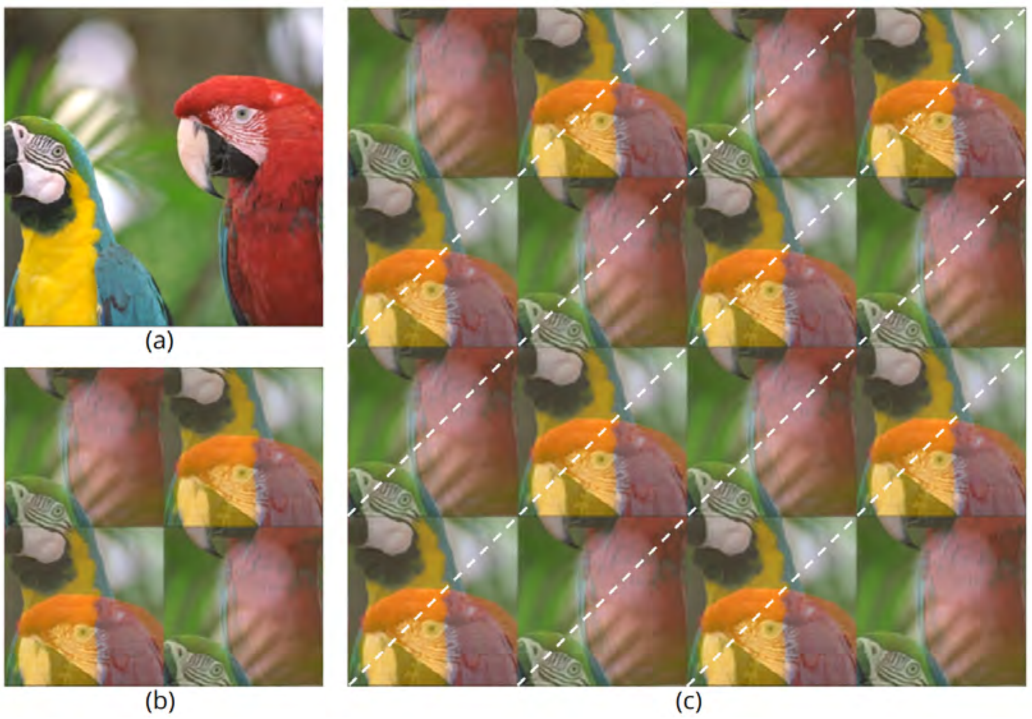

Figure 7: (a) Ground truth image. (b) A faulty representation is given by a model initialized with only odd input frequencies. (c) Image extrapolation showcasing a diagonal periodic pattern.

The network depicted in Figure 7 considered odd frequencies, but similar phenomena occur in other cases. This includes scenarios where ω is composed of even frequencies (Figure 8(a)), or exclusively positive ones (Figure 8(b)). Specifically, in the case where the input frequencies are odd, and the network period is 3 (Figure 8(c)), we observe an overlap of images, confirming that the issue arises from the incongruence between the period of the network p and the period generated by the input frequencies.

To obtain a rigorous classification of the set of initializations with sub-period generation, let's consider the hidden neuron $h_i(\mathbf{x})$ as the i-th coordinate of vector $\mathbf{S}_{\mathbf{A},\mathbf{b}} \circ \mathbf{H}(\mathbf{x})$. Then, using Theorem 1 it's obtained the expression

$$h_i(\mathbf{x}) = \sum_{\mathbf{k}\in\mathbb{Z}^n} \alpha_{\mathbf{k}}(\mathbf{A}_i)\sin(g(\mathbf{x},\theta))$$

where

$$g(\mathbf{x},\theta) = \left\langle \mathbf{k}, \frac{2\pi}{p}\omega^1 \right\rangle x + \left\langle \mathbf{k}, \frac{2\pi}{p}\omega^2 \right\rangle y + \langle \mathbf{k}, \varphi \rangle + b,$$

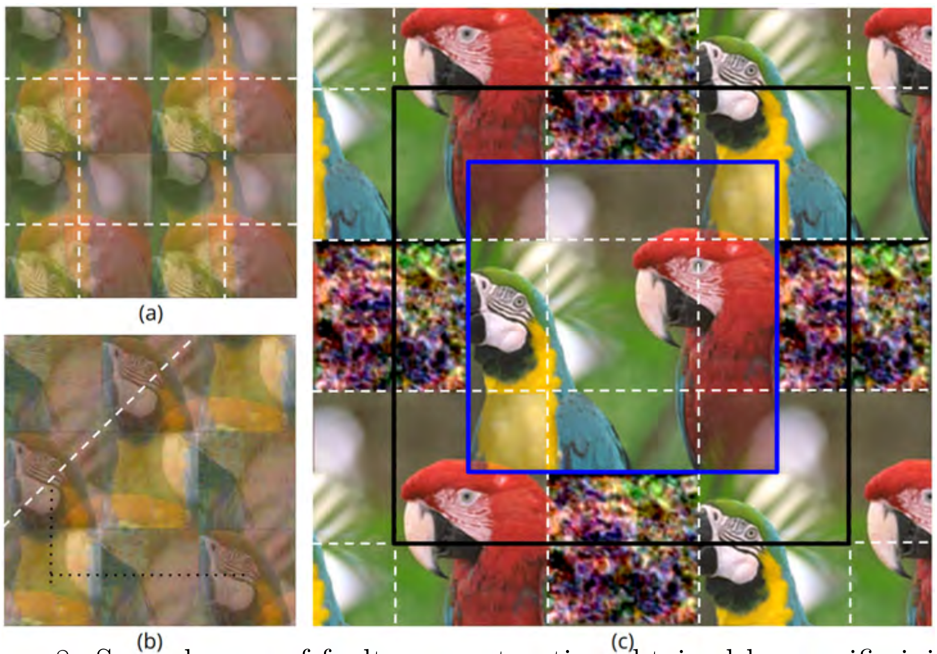

Figure 8: Several cases of faulty reconstruction obtained by specific initializations of ω_0. (a) Reconstruction from a model with even input frequencies. (b) Reconstruction with $\omega_0 = [(k,l) : k,l \in \{1,10,100\}]$. (c) Extrapolation of reconstruction with odd input frequencies and period $p = 3$. The black box determines the injective domain of the 3-periodic function and the blue box demarcates the image domain.

Let ω^j represent the j-th column of ω, and let $\mathbf{x} = [x,y]$. For simplicity, we will omit the index i from h_i. Note that h is a periodic function with a period of p. However, there is no assurance that a sub-period will not emerge.

If h has a sub-period then $h(x,y) = h(x + \frac{p}{q}, y + \frac{p}{l})$ with $p/q, p/l$ sub-periods in axis X and Y, respectively. Now,

$$h\left(x + \frac{p}{q}, y + \frac{p}{l}\right) = \sum_{\mathbf{k}\in\mathbb{Z}^n} \alpha_{\mathbf{k}}(\mathbf{A}_i) \sin\left(g(\mathbf{x},\theta) + 2\pi\left\langle \mathbf{k}, \frac{\omega^1}{q} + \frac{\omega^2}{l}\right\rangle\right)$$

then, h (and consequently f_θ) manifests sub-periods when $\left\langle \mathbf{k}, \frac{\omega^1}{q} + \frac{\omega^2}{l}\right\rangle \in \mathbb{Z}$ for all $\mathbf{k} \in \mathbb{Z}^n$. A visual test can detect a sub-period problem, and it is enough to introduce a new input frequency such that $\left\langle \mathbf{k}, \frac{\omega^1}{q} + \frac{\omega^2}{l}\right\rangle \notin \mathbb{Z}$ for some $\mathbf{k} \in \mathbb{Z}^n$ to solve it.

The second problem arises when questioning the need to choose the initial frequencies for each signal. Figures 3 and 6 show that it is possible to learn

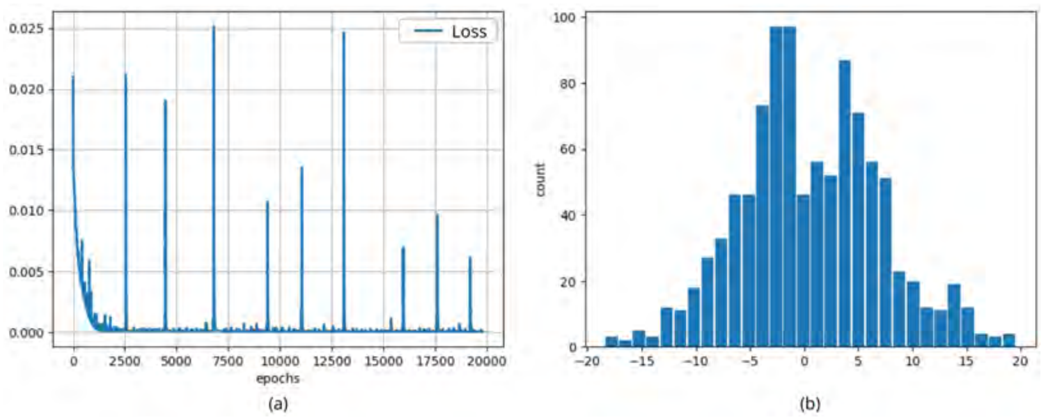

Figure 9: Graphics obtained when fitting the signal $f(x) = \sum_{k=1}^{100} \sin(k\pi x)$ with $\omega = \mathbf{1}$ and ω_0 (as defined by [27]) adjusted to 1000. (a) Loss evolution during training. (b) Histogram of the learned network's hidden matrix $\mathbf{A}$.

a signal by initializing $\omega = \mathbf{1}$ and appropriately setting ω_0. However, this parameter strongly affects the stability of the training and tends to introduce noise into the reconstructions.

Figure 9(a) reflects the instability even in advanced stages of training. To explain this phenomenon, observe the weight distribution of the hidden matrix $\mathbf{A}$ (see Figure 9(b)). The values of $\mathbf{A}_i$ are far from the origin, then the terms $\alpha_{\mathbf{k}}(\mathbf{A}_i) = \prod_{j=1}^{n} J_{k_j}(A_{ij})$ may become significant for $|\mathbf{k}|_\infty >> 1$. Therefore, the neural network uses high order frequencies to fit the signal, which can cause artifacts or overfitting.

Furthermore, values greater than 1.4 can be roots of first-order Bessel functions. Therefore, if A_{ij} is the root of a Bessel function J_l, all the generated frequencies $\langle \mathbf{k}, \omega \rangle$ such that $k_j = l$ will have zero coefficient in that epoch. The disappearance of an infinite number of frequencies and subsequent appearance when updating the value of A_{ij} is most likely one of the main factors destabilizing training.

Consequently, when encountering a training instability problem using low initial frequencies and a high value of ω_0, it is advisable to reduce ω_0 and increase the number or bandwidth of the initial frequencies.

Note that allowing a high value of $\|\mathbf{A}\|_\infty$ generated various problems during training, so controlling the range of values that $\mathbf{A}$ can take could influence learning positively. Indeed, the following chapter will present a scheme that restricts the growth of $\mathbf{A}$, obtaining stability and control of the network's bandwidth.

6 Bounding

According to Theorem 2, the coefficients of the generated frequencies are bounded by an expression dependent on the norm of the rows $\mathbf{A}_j$ of $\mathbf{A}$. In particular, Equation 9 shows that by limiting the infinite norm of $\mathbf{A}$ by a sufficiently small value, higher order frequencies become negligible.

For example, considering a neural network with matrix $A \in \mathbb{R}^{5\times1}$ such that $\|\mathbf{A}\|_\infty < 1$ and $|k|_\infty > 5$, the Theorem 2 indicates that $|\alpha_k(\mathbf{A}_j)| < \prod_{i=1}^{m} \frac{1}{2^{|k_i|}|k_i|!} \leq \frac{1}{2^5 5!} \approx 2.604e^{-4}$. The previous example exhibits that frequencies at or above order 5 exert reduced impact on the representation.

Thus, by limiting $\mathbf{A}$, it is possible to control the frequency band of the reconstruction so that the generated frequencies saturate around small multiples of the input frequencies. We call this process as *bounding.*

Definition 6.1 (Bound)**.** *Given a matrix $\boldsymbol{A}$, the bound of $\boldsymbol{A}$ is defined as the matrix with entrances $f_B(A_{ij})$ where $B \in \mathbb{R}_+$ and*

$$f_B(x) = \begin{cases} \max(x, B) & if \quad x \leq 0 \\ \\ \min(x, B) & if \quad x > 0 \end{cases}$$

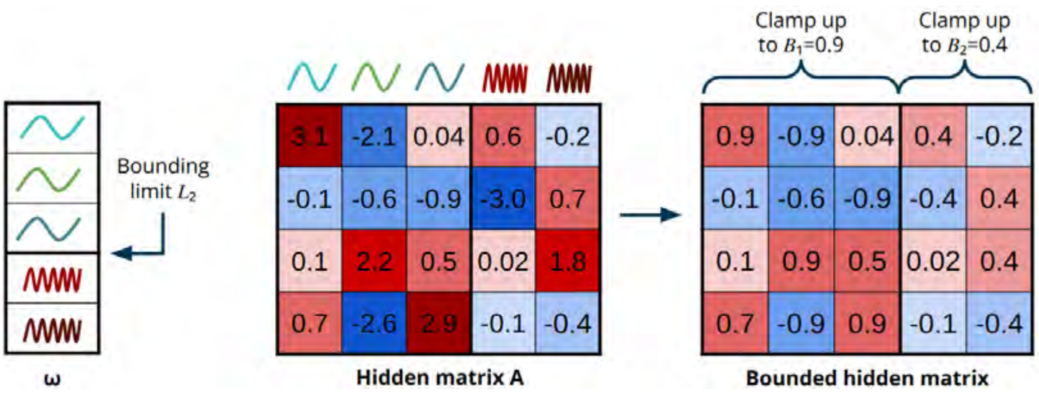

Figure 10: Illustrative example of the bounding process considering a partition of ω in low and high frequencies.

We establish a training scheme that bounds the hidden matrix $\mathbf{A}$ after each gradient descent step. Figure 10 shows a toy example of the bounding process in a 4×5 hidden matrix $\mathbf{A}$ with two blocks.

The constant B becomes a hyperparameter that limits $\mathbf{A}$, and from Figure 11, we can appreciate the bandwidth control of the representation that it exerts. Observe that with constant bounds, the spectral distribution of the reconstructed model considers all initial frequencies having the same importance over it. However, this is seldom true for images, as it's known that most energy concentrates around low frequencies of the spectrum.

Based on this fact, we consider restricting each input frequency based on its possible impact on the reconstruction. Based on the spectral bias phenomena, we partition the model spectrum as shown in Figure 19, defining $\mathbf{L}$ limits of the partition and $\mathbf{B}$ a decreasing vector of bounds as new hyperparameters to finetune.

Formally, we define a restriction of the norm to the columns of $\mathbf{A}$ according to the frequency with which they are associated. The above is motivated by the following example: Let $f_\theta : \mathbb{R} \to \mathbb{R}$ be a sinusoidal neural network such that $\omega = [1, 100]$, then the generated frequencies have the form $k \cdot 1 + l \cdot 100$. By allowing $|k| >> 1$, the generated frequencies will be more likely to be present in the image frequency band. On the other hand, allowing $|l| >> 1$ indicates a greater risk of creating frequencies outside the Nyquist limit, introducing noise.

Formally, consider the vectors $\mathbf{L} \in [0, \infty)^l \times \{\infty\}$, $\mathbf{B} \in [0, \infty]^l$ with $\mathbf{L}$ sorted increasingly and $L_1 = 0$. Then, *block bounding* can be defined as the process of bounding each column $\mathbf{A}^j$ of $\mathbf{A}$ with hyperparameter defined as the B_i such that the frequency ω_j related to $\mathbf{A}^j$ satisfy

$$\max(|\omega_j^1|, |\omega_j^2|) \in [L_i, L_{i+1}), \quad i \in \{1, ..., l\}.$$

As the coordinates L_i for $i \in \{2, ..., l\}$ completely determine $\mathbf{L}$, we simplify notation by defining $\mathbf{L}$ only through those values. Figure 10 illustrates the block bounding process with $l = 2$.

When applying bounding with parameter $B_i = \infty$ for some i, it's equivalent to the default training (no bounding) for the weights on that block.

Table 13 presents the ssim metrics when considering several values for $\mathbf{B}$ and $\mathbf{L}$, and $\omega_0 = 30$ for the kodak monumentum image. Observe that quality over the reconstruction worsens when selecting small limits for low frequencies, and the best metrics consider $L_3 = 20$.

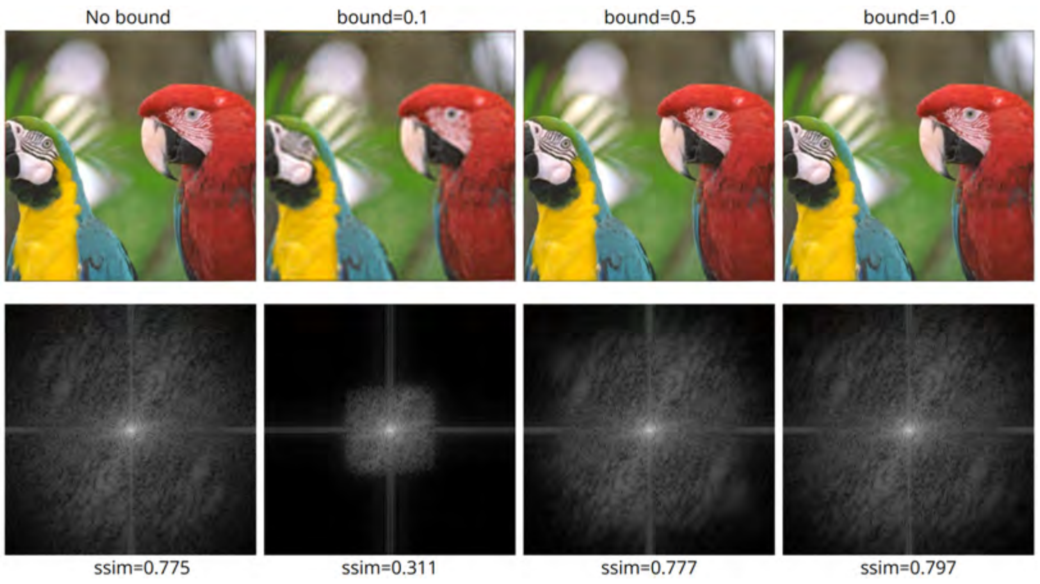

Figure 11: Fitting the image during 3000 epochs with different bound values. (Above) Reconstructions obtained from a training bounding the hidden matrix **A**. (Below) Spectrum of the images above.

To understand this phenomenon, observe the initial frequencies' distribution Figure 12 (b) that shows a higher frequency count in values over 26. Then, as bounds limit the high frequencies, the model restricts a large quantity of initial atoms from generating new frequencies. Indeed, see in Figure 13 that instead of introducing artifacts, the reconstruction done with $\mathbf{B} = [1.3, 0.6, 0.05]$ and $\mathbf{L} = [5, 10]$ which presents one of the worst reconstruction has a similar appearance to a softened version of the image, showing promise as a filtering technique for sinusoidal neural representations.

As the spectral partition defined imposes a non-uniform distribution of input frequencies, we consider different initializations for the initial weights more aligned with the techniques used in this work.

6.1 Initialization and Bounding

We consider initializing the model with frequencies concentrated along the first bounding limit. Then, as shown in Figure 14, besides having a visibly higher quality, the generated frequencies appear around initialized frequencies, corroborating the bias of learning low order frequencies first.

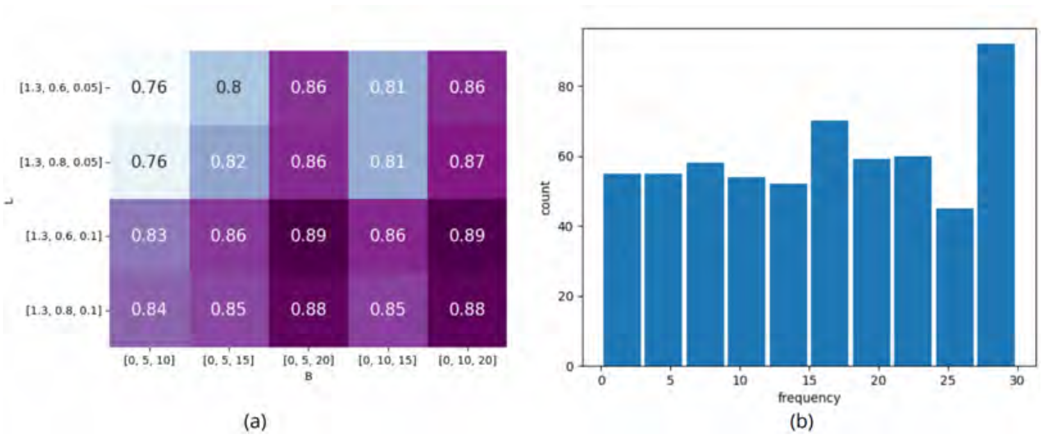

Figure 12: (a) Table of SSIM metrics of trained models using different values for **B** and **L**. (b) Histogram of frequencies (in absolute value) appearing along the axes X and Y.

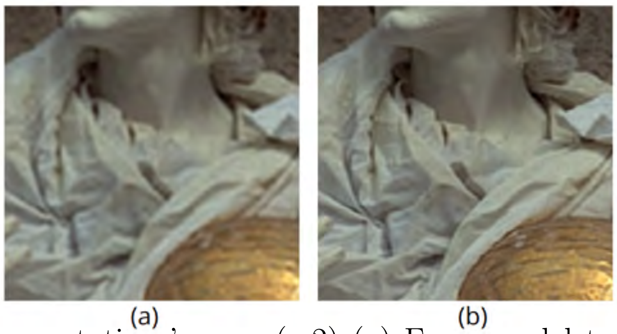

Figure 13: Representations' zoom ($\times 2$) (a) For a model trained with $\mathbf{B} = [1.3, 0.6, 0.05]$ and $\mathbf{L} = [5, 10]$. (b) For a network with $\mathbf{B} = [1.3, 0.6, 0.1]$ and $\mathbf{L} = [5, 20]$.

Then, by distributing the frequencies with symmetry as shown in Figure 14(d), the spread of frequencies along the representation's Fourier spectrum is more homogeneous, becoming a good initialization for all sorts of images.

Finally, observe that unlike the diffusion scheme, where sometimes stopping the training earlier renders higher quality images by avoiding undesirable overfitting, the noise appears from the beginning of our training. Figure 14(a)-(b) shows that this phenomenon of high-frequency noise happens very early in the training. Then, most of the training steps focus on denoising the reconstruction.

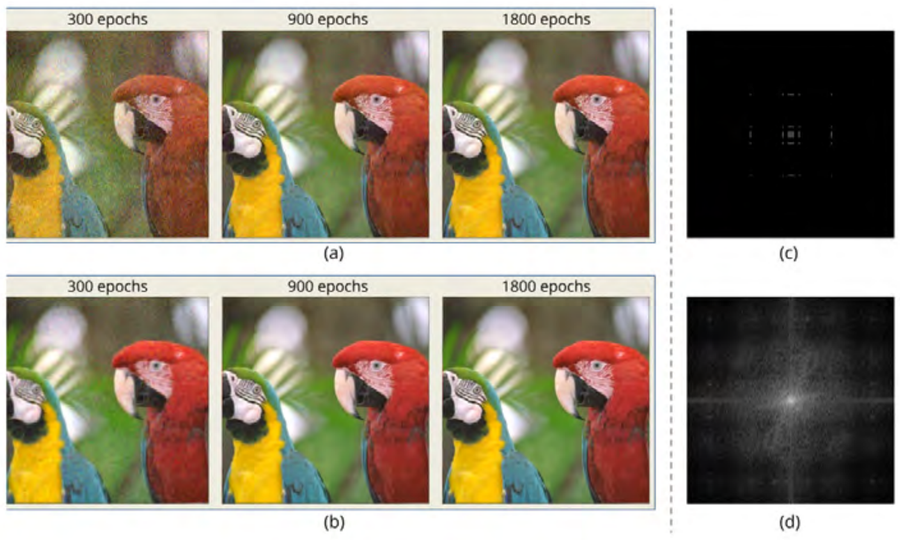

Figure 14: Reconstruction's evolution during training where $\omega = \{(i,j) : i,j \in \{1,-1,3,-3,10,-10,50,-50\}\}$ and Nyquist limit is $\Omega = 128$. (a) Train with Sitzmann et al. [27] scheme. (b) Train with bounding scheme. (c) Distribution of input frequencies along the domain $\mathbb{Z}^2$ where the center of the image corresponds to frequency $(0,0)$. (d) Spectrum of a network trained using bounding during 300 epochs.

7 Compression

While employing the mean squared error loss function, $\mathcal{L}_{MSE}$, in the training process, there is no assurance that the model is learning without introducing redundancy. Indeed, Johnson [11] presented in his lecture an illustrative example in which the elements of the last sinusoidal layer are highly correlated. As these components will be linearly combined to form the output of the function, it is reasonable to question whether there is a better scheme capable of decorrelate each output atom, and if so, whether it is possible to perform the same task with a network with a smaller number of parameters. This hypothesis is also known by the name of Lottery Ticket Hypothesis [7, 9].

This chapter explores a method to mitigate the redundancy inherent in sinusoidal networks. This approach examines the initialization method described earlier, which treats the elements of the first layer as atoms of the Fourier basis. Drawing inspiration from the works of Johnson [11] and Benbarka et al. [1, section 3.4], we implement a weight decay scheme (refer to Definition 3.2) coupled with a structured pruning process. This combined strategy effectively trims down the network size, mitigating accuracy loss.

7.1 Method

The sinusoidal neural network fits the signal using an optimization algorithm that seeks to minimize the loss functional $\mathcal{L}_{MSE}$. Then, by considering a new functional,

$$\mathcal{L}_{reg} = (1-\alpha)\mathcal{L}_{MSE} + \alpha\mathcal{L}_{wd}, \qquad \alpha \in [0,1]$$

as the target function to minimize, the network will train to reconstruct the image conditioning the model by applying a *regularization* on the parameters, introduced by $\mathcal{L}_{wd}$. This term will induce the values of $\mathbf{A}$ to be close to zero, gathering most of the reconstruction's energy around low order generated frequencies.

Formally, let $f_\theta = \mathbf{CS}_{\mathbf{A},\mathbf{b}} \circ \mathbf{H}$ be a fully trained neural network with $\mathbf{A} \in \mathbb{R}^{n\times m}$, $e_{wd}, e_{rec}, e_{total} \in \mathbb{N}$ the number of epochs to train with weight decay ($\alpha \neq 0$), with the usual loss ($\alpha = 0$) and in total, respectively, and $T_p \in [0,1]$ a *pruning threshold* that indicates how much information can lose per pruning. The pruning scheme is given by,

1. Train f_θ during e_{wd} epochs with loss functional $\mathcal{L}_{reg}$.
2. Select a subset $\mathcal{I} \subseteq \{1,...,n\}$ of row (column) indices based on the criteria: The greatest subset of $\{1,...,n\}(\{1,...,m\})$ such that $\sum_{i\in\mathcal{I}} \|\mathbf{A}_i\|_1 < T_p$ ($\sum_{i\in\mathcal{I}} \|\mathbf{A}^i\|_1 < T_p$). Prun the rows (columns) with those indices.
3. Fine-tune f_θ using the loss functional $\mathcal{L}_{MSE}$ during e_{rec} training steps.
4. Consider the stop criteria: The number of epochs elapsed $e \geq e_{total}$ or $\mathcal{I} = \emptyset$. If met the stopping criteria, stop the process. Otherwise, return to step 1.

Before testing the method, it's crucial to note that Chapter 4 delved into the impact of sinusoidal neural network weights on the reconstruction process. This understanding allows us to anticipate some effects that pruning might induce on the reconstruction.

Observe that two coordinates of the sinusoidal layer $\mathbf{S}_{\mathbf{A},\mathbf{b}} \circ \mathbf{H}(\mathbf{x})$ are infinite linear combinations of harmonics with equal frequencies but each with different coefficients. Therefore, pruning a row of $\mathbf{A}$ affect exclusively the coefficients $\widehat{A}_{\mathbf{k}}$ and $\widehat{B}_{\mathbf{k}}$, and thus the configuration of generated frequencies.

On the other hand, we nullify all terms associated with ω_i when pruning the i-th column of $\mathbf{A}$. Therefore, in the harmonic expansion of Theorem 1, all generated frequencies with non-zero coefficient in the i-th component ($\langle \mathbf{k}, \omega \rangle$ with $k_i \neq 0$) will disappear. Therefore, it's possible to lose generation capacity as the model doesn't have access to the same set of frequencies to represent the image.

In that sense, pruning columns agree with matching pursuit theory, as the process naturally selects those initial harmonic atoms $\mathbf{H}_i$ that exert the most influence over the reconstruction.

7.2 Experiments

During the pruning process, several hyperparameters directly impact the quality of the reconstruction. We conducted an ablation study concerning the following values: the regularization parameter α, the norm used in the weight decay functional, and the effect of cutting rows, columns, or both during the pruning process. The proposed method is tested on the Kodek dataset to analyze its utility, and a variation of the scheme considering the mathematical approach to the problem is presented.

7.2.1 Regularization parameter α

During the pruning process, the parameter α measures the influence of the regularization term on training. Intuitively, α sets the amount of information corrupted to reduce the model.

To find suitable values for α, let's assume that $\alpha = \alpha_s \cdot \alpha_f$, where $\alpha_f : \mathbb{N} \to [0, 1]$. Thus, it suffices to find the appropriate function scale and subsequently test the training behavior with different definitions of α_f. We seek an α_s that offers the best trade-off between weight reduction of the matrix and SSIM of the reconstructed image.

When training a model using weight decay with $\alpha = 1e-5, 1e-6, 1e-7,$

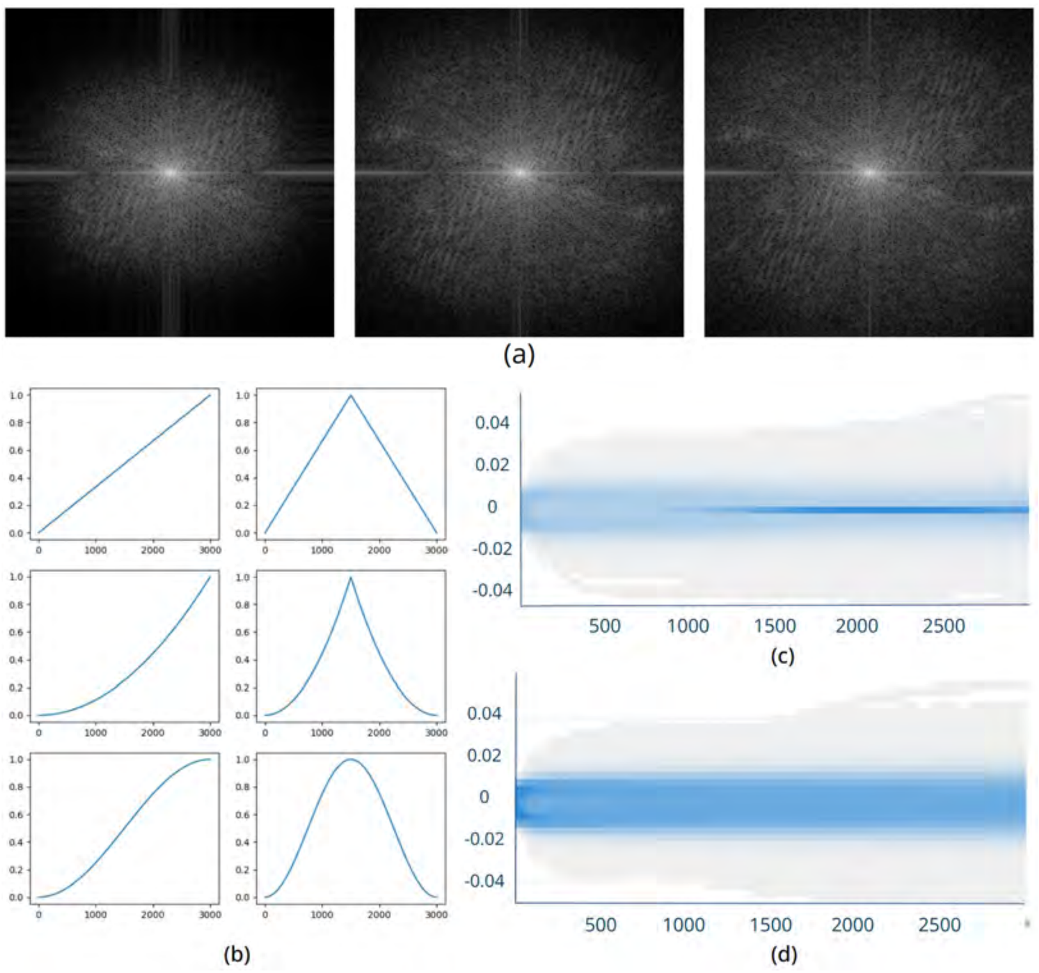

Figure 15: (a) Fourier spectrum after training using weight decay with constant $\alpha = 1e-5, 1e-6, 1e-7$, respectively. (b) Candidates for α_f. (c) Histogram over training epochs (where intensity measures the number of weights in the range given by y) when using $\mathcal{L}_{wd}$ with $\alpha = 1e-6$. (d) Histogram over training epochs with $\alpha = 1e-5$.

the following similarity (SSIM) values are obtained: 0.88, 0.79, and 0.58. Since the SSIM metric of the original trained model is given by 0.88, $\alpha_s = 1e-7$ seems like an ideal value. However, note in Figure 15 (d) that at this scale, the weights distribution of $\mathbf{A}$ disperses from the origin during training. On the other hand, Figure 15 (c) shows the weights of $\mathbf{A}$ accumulating at the origin when training with $\alpha = 1e-6$, and thus we consider $1e-6$ as a reasonable value for α_s.

Consider the function α_f from one of these two families of functions: increasing functions and piecewise functions defined as

$$g(x) = \begin{cases} g_1(x) & \text{if } x \leq \frac{e_{wd}}{2} \\ g_2(x) & \text{if } x > \frac{e_{wd}}{2} \end{cases}$$

where $g_1, -g_2$ are increasing functions (see Figure 15 (b)). The studies conducted showed that the functions of the second class manage to reduce the norm of the matrix $\mathbf{A}$ and achieve a quality closer (on the scale $1e-5$) to a representation obtained with the default training scheme compared to increasing functions (scale $1e-4$). Then, we consider $\alpha_f = g$, where g is a piecewise function as above with g_1, g_2 being quadratic functions where $g_1(0) = 0, g_1(\pm\frac{e_{wd}}{2}) = 1$, and $g_2(e_{wd}) = 0, g_2(e_{wd} \pm \frac{e_{wd}}{2}) = 1$, as it showed the most promising results among the second class.

Note that the principle behind weight decay shares some similarity with bounding since both reduce the range of values that the weights of $\mathbf{A}$ can take. Indeed, Figure 15 (a) shows the limiting effect of α on the spectrum of the reconstruction. However, as seen earlier, weight decay has a more detrimental repercussion on the representation's quality than bounding.

7.2.2 Weight decay norm

When defining weight decay, we choose the $L1$-norm as a proxy for the $L0$-norm, following a similar scheme as the one proposed in compressive sensing theory. A pruning process is applied to each image in the Kodak dataset (using the same seed), and we compare the accuracies of the respective reconstructions. The experiments show that no norm has a significant advantage over the other. Therefore, the experiments using weight decay will use $L1$-norm by default.

7.2.3 Pruning components

The rows and columns of the network play different roles in the reconstruction process. To measure the impact of reducing one of these components, let's consider an algorithm that targets only the rows, only the columns, and rows and columns alike.

Figure 16 shows the average accuracy obtained by reducing over 50% of the neural network parameters. It indicates that cutting lines is more

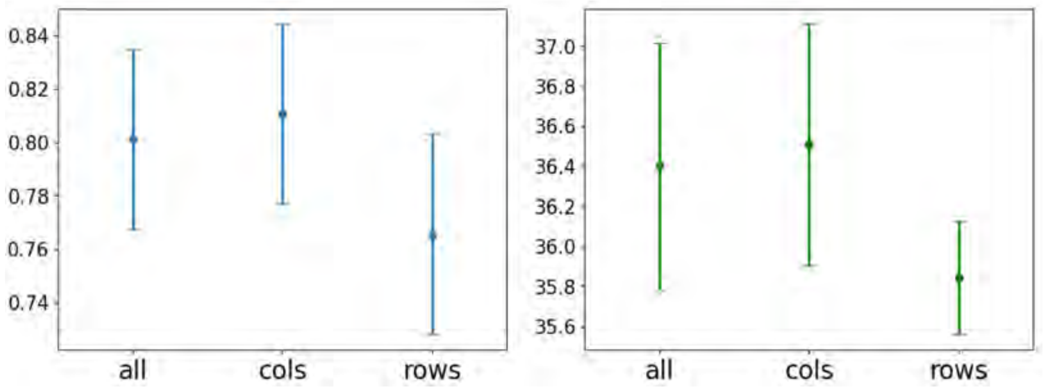

Figure 16: Accuracy with confidence intervals of the pruning method. The scheme pruned over 50% of the columns, rows, or both of a network f_θ with $\mathbf{A} \in \mathbb{R}^{300\times 300}$ during 100000 epochs, and the confidence intervals were obtained using kodak dataset. (Left) SSIM. (Right) PNSR.

detrimental to reconstruction than the other two schemes. While it is true that reducing weights along rows to values near zero is easier and introduces fewer instabilities at the beginning of training, the model achieves better fine-tuning when eliminating columns.

The previous results show that by allowing the network to implicitly choose the architecture (rows-columns ratio) that provides greater representation capacity, a hidden matrix such that $n > m$ is preferred. In the context of matching pursuit, the model is more favorable to reconstructing using many simple functions (output atoms) than a few complex functions.

On the other hand, Figure 17 shows a comparative accuracy loss when pruning models with different initialization settings. As seen in the graphic, our pruning method allows us to reconstruct the image with more than 70% accuracy even when pruning 75% of the weights of $\mathbf{A}$. It's worth noting that even after manually selecting the initial frequencies, the model retained the structure obtained from setting the frequencies to integers. As a result, it performed nearly identically to the case with random initialization.

Moreover, we compare pruning a model with our initialization method and pruning a model with the initialization scheme defined by [27]. When pruning 25% of weights, our method ensures almost no loss in quality (at least 99.8%), whereas SIREN exhibits a reduction in accuracy (87.7%).

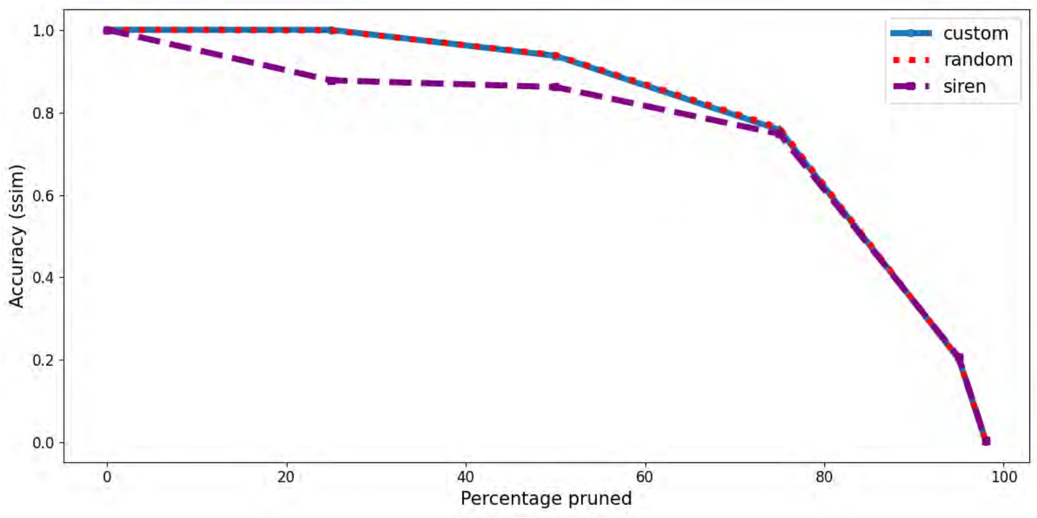

Figure 17: Accuracy loss due to pruning, where the accuracy is relative to the pre-pruned reconstruction. 'custom' refers to a hand-picked initialization, 'random' is a random choice of initial weights and siren using the first layer initialization defined in [27].

7.2.4 Targeted weight decay

Values of α close to 1 induce the matrix $\mathbf{A}$ to approach zero, and increasing the influence of the regularization parameter further degenerates the representation, trending towards the constant zero function. This argument highlights an issue with the previously used weight decay, as the objective is not to obtain a representation with null $\mathbf{A}$ but to zero out the values of $\mathbf{A}$ with less significance.

Therefore, consider the following modification to the algorithm presented above,

1. Train f_θ during e_{wd} epochs with loss functional $\mathcal{L}_{reg}$.
2. Select a subset $\mathcal{I} \subseteq \{1, ..., n\}$ of row (column) indices based on the criteria: The greatest subset of $\{1, ..., n\}(\{1, ..., m\})$ such that $\sum_{i\in\mathcal{I}} \|\mathbf{A}_i\|_1 < T_p$ $(\sum_{i\in\mathcal{I}} \|\mathbf{A}^i\|_1 < T_p)$.
3. Apply targeted weight decay in rows (columns) over the set $\mathcal{I}$ during e_{twd} epochs.
4. Prun the rows (columns) $\mathbf{a}$ such that $\|\mathbf{a}\| < T_n$.

5. Fine-tune f_θ using the loss functional $\mathcal{L}_{MSE}$ during e_{rec} training steps.

6. Consider the stop criteria: The number of epochs elapsed $e \geq e_{total}$ or $\mathcal{I} = \emptyset$. If the stop criteria is reached, stop the process, otherwise return to step 1.

where $e_{twd} \in \mathbb{N}$ and $T_n \in [0, \infty]$. Note that under an appropriately chosen set of parameters, this algorithm does not account for the degenerate solution $f_\theta \equiv 0$, even in the extreme case of $\alpha = 1$.

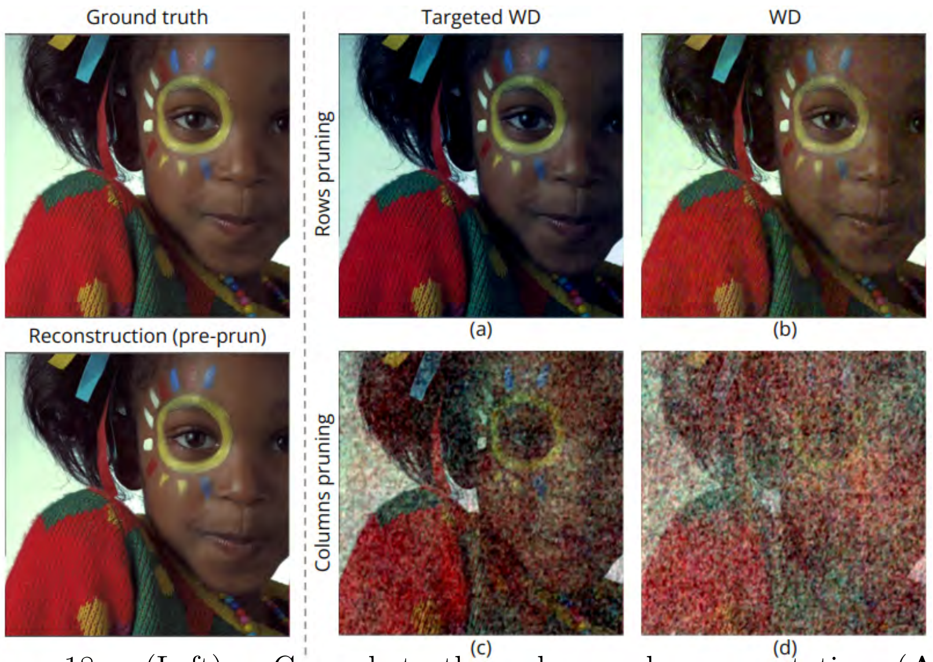

Figure 18: (Left): Ground truth and neural representation ($\mathbf{A} \in \mathbb{R}^{800\times200}$). (Right): Effect of structured pruning of 10% of representation's rows/columns without fine-tuning. (a): Row pruning after applying targeted weight decay maintains structure while losing intensity. (b): Row pruning after training with weight decay keeps intensity but introduces artifacts. (c) & (d): Pruning columns greatly impacts the representation.

In step 2, the model identifies the rows (columns) of the network that appear to contain the least information. Subsequently, in step 3, weight decay guides the training to attempt to nullify these weights. If any of these rows (columns) are indeed essential, step 4 ensures they are not removed.

More formally, let's consider that the initial approximation derived from the reconstruction is defined in a space of dimension n. Intuitively, one might propose that during the weight decay process, the goal is to find an approximation closer to the origin of that space. Conversely, targeted weight decay seeks a solution belonging to a subspace of that space with a dimension $m < n$.

In Figure 18, the distinction between applying and not applying targeted weight decay is evident. For the case of lines, it is observable that pruning rows only impacts the intensity of the image. In the standard scenario, artifacts typical of sinusoidal networks are introduced. On the other hand, it is noticeable that targeted weight decay preserves more information compared to weight decay when pruning columns.

Given the insights from the preceding observations, the next chapter will delve into pruning, considering all the aforementioned refinements to the algorithm.

8 Compression and Initialization

When pruning a sinusoidal neural network, the matching between the defined ω and the image's dominant frequencies is an important factor in guaranteeing good quality with fewer parameters. This chapter considers the problem of finding suitable dictionary atoms to obtain a compact representation of the model.

Using the bounding training scheme it's possible to define a partition of $\mathbf{A}$'s columns into *blocks* as

$$\mathbf{A}_{(i)} := \{\mathbf{A}^j : \omega_j \in [L_i, L_{i+1})\}.$$

where follows from the definition that $\left\|\mathbf{A}_{(i)}\right\|_\infty < B_i$.

When a weight A_{ij} from $\mathbf{A}$ has saturated to $\pm B_k$, Theorem 1 indicates that its related frequency ω_j is used to introduce as many generated frequencies as the bound allows. Then, having a block $\mathbf{A}_{(k)}$ with many weights that saturated to $\pm B_k$ could signal a strong relationship between the range of input frequencies $[L_k, L_{k+1})$ and the image Fourier spectrum.

We use this concept of saturation over each block to measure the dependence of the reconstruction over the frequencies in each region of the spectrum.

Definition 8.1. *Given a closeness tolerance $\epsilon > 0$, the importance criteria I_k that characterizes the reconstruction dependence over the block $\boldsymbol{A}_{(k)}$ is the value*

$$I_k = \sum_{w \in \boldsymbol{A}_{(k)}} B_k \frac{\max\{|w| - B_k + \epsilon, 0\}}{|\mathcal{B}|_i}$$

The criterion is a weighted measure of the values of $\mathbf{A}$ that saturate to the bound. Refer to Figure 19 for a toy example of column selection criteria.

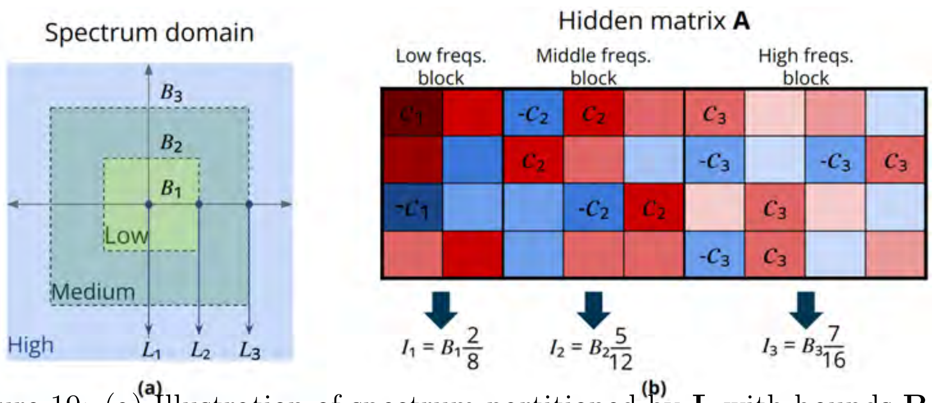

Figure 19: (a) Illustration of spectrum partitioned by $\mathbf{L}$ with bounds $\mathbf{B}$. (b) Toy example of the criteria used to add columns.

Given the importance criteria defined by $\mathbf{B}$ and $\mathbf{L}$, $n_{add}, N_{add} \in \mathbb{N}$ the number of columns to add at once and in total (with $n_{add}|N_{add}$), $e_{rec}, e_{total} \in \mathbb{N}$ the number of epochs to train before adding a column and to train in total, and $\epsilon > 0$ a closeness tolerance, we define the algorithm as follows:

1. Train f_θ during e_{rec} epochs.
2. Compute I_k for each block $\mathcal{B}_k$.
3. For $I_s = \max\{I_k : k \in \{1, ..., l\}\}$, add n_{add} new columns and link them to frequencies in the range $[L_s, L_{s+1}) \times [0, L_{s+1}) \cup [0, L_{s+1}) \times [L_s, L_{s+1})$. If there are two maximum criteria I_{s_1} and I_{s_2}, select the one with the lower s_i.
4. If the total number of columns (epochs) added up this point is equal to N_{add} (e_{total}), stop iterations. Otherwise, go back to step 1.

Observe in Figure 20 that an image trained with a gradual increase of input frequencies outperforms a model fitted with a uniform spectrum restricted to an ill-fitted bandwidth. In such instances, it is preferable to control the generated frequencies rather than train with a network with more parameters.

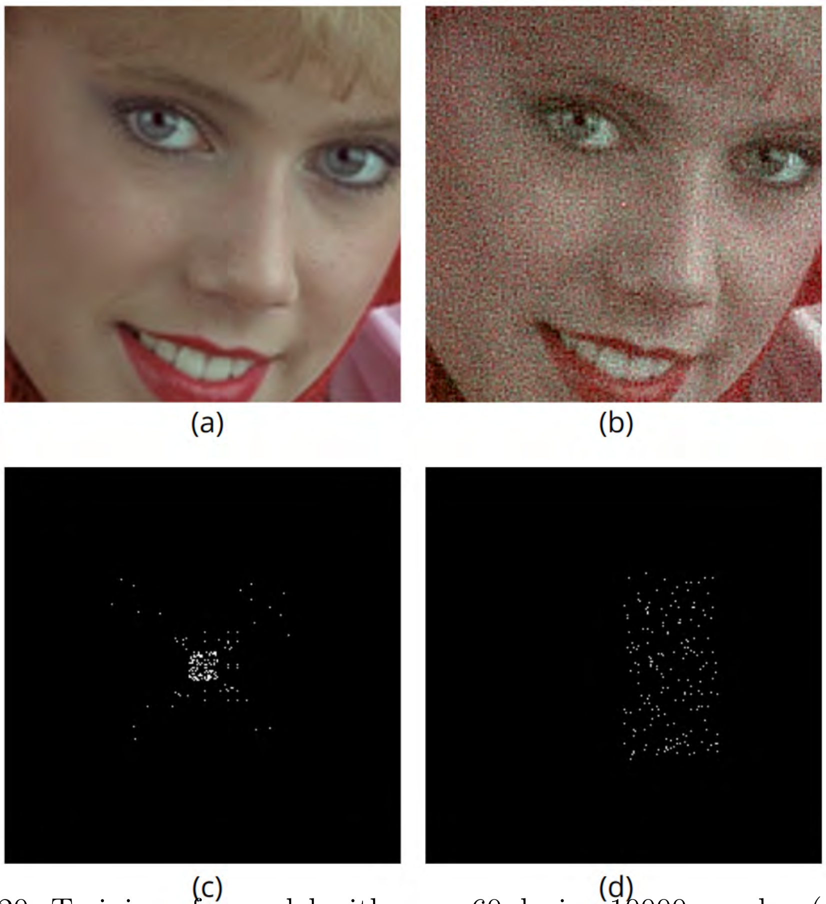

Figure 20: Training of a model with $\omega_0 = 60$ during 10000 epochs. (a) Zoom in of reconstruction obtained from an adding columns' scheme with starting hidden matrix $\mathbf{A} \in \mathbb{R}^{18\times300}$ and final hidden matrix $\mathbf{A} \in \mathbb{R}^{178\times300}$. (b) Zoom in on the reconstruction with base training. (c) Input frequencies of the final model trained with adding columns' scheme. (d) Input frequencies randomly selected from $\mathbb{Z}^2$ with bandwidth $\omega_0 = 60$.

Moreover, the criterion selects frequencies in line with spectral bias, choosing low ones at the beginning of training and selecting higher ones when necessary. As such, 20(c) shows that selected frequencies center the distribution of frequencies around the origin.

9 Conclusions

This work proposed several methods to improve the initialization, training, and efficiency of sinusoidal neural networks. First, we defined an initialization strategy that provided the mathematical ground to work the following methods. Then we introduced a bandwidth control scheme that offered higher quality representations with no additional training cost, which was used to create an architecture search algorithm. Finally, we studied compression over sinusoidal neural networks using structured pruning, exploiting the framework given by our initialization.

Our contributions aimed to present strategies that could solve problems present in sinusoidal networks such as noisy reconstructions in deep (sinusoidal) models or training instabilities. As such, future directions could consider studying the implications of our techniques in deep models or further developing the theoretical framework by using diophantine equations and probability theory.

Another future work would aim towards a complete compression pipeline to transform sinusoidal neural networks in real-time, high-quality image representations.

References

[1] Nuri Benbarka, Timon Höfer, Andreas Zell, et al. Seeing implicit neural representations as fourier series. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2041–2050, 2022.

[2] Yinbo Chen, Sifei Liu, and Xiaolong Wang. Learning continuous image representation with local implicit image function. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8628–8638, 2021.

[3] Megh Doshi, Varun Sundar, and Zachary Huemann. Cs766 project: Neural image compression. 2021.

[4] Yishun Dou, Zhong Zheng, Qiaoqiao Jin, and Bingbing Ni. Multiplicative fourier level of detail. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1808–1817, 2023.

[5] Emilien Dupont, Adam Golinski, Milad Alizadeh, Yee Whye Teh, and Arnaud Doucet. Coin: Compression with implicit neural representations. In *International Conference on Learning Representations*, 2021. URL `http://arxiv.org/abs/2103.03123v2`.

[6] Emilien Dupont, Hrushikesh Loya, Milad Alizadeh, Adam Goliński, Yee Whye Teh, and Arnaud Doucet. Coin++: Neural compression across modalities. *arXiv preprint arXiv:2201.12904*, 2022.

[7] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pages 2943–2952. PMLR, 2020.

[8] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[9] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=rJl-b3RcF7`.

[10] Song Han, Han Cai, and Ji Lin. Mit 6.5940 fall 2023 tinyml and efficient deep learning computing, 2023. URL `https://hanlab.mit.edu/courses/2023-fall-65940`.

[11] Matthew Johnson. Fourier feature networks and neural volume rendering. URL `https://www.youtube.com/watch?v=cXoaCw796Do`.

[12] David B Lindell, Dave Van Veen, Jeong Joon Park, and Gordon Wetzstein. Bacon: Band-limited coordinate networks for multiscale scene representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16252–16262, 2022.

[13] Gaurav Menghani. Efficient deep learning: A survey on making deep learning models smaller, faster, and better. *ACM Computing Surveys*, 55(12):1–37, 2023.

[14] 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. *Communications of the ACM*, 65(1):99–106, 2021.

[15] Tiago Novello. Understanding sinusoidal neural networks. *arXiv preprint arXiv:2212.01833*, 2022.

[16] Tiago Novello, Guilherme Schardong, Luiz Schirmer, Vinicius da Silva, Helio Lopes, and Luiz Velho. Exploring differential geometry in neural implicits. *Computers & Graphics*, 108:49–60, 2022.

[17] Tiago Novello, Vinicius da Silva, Guilherme Schardong, Luiz Schirmer, Helio Lopes, and Luiz Velho. Neural implicit surface evolution. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14279–14289, 2023.

[18] Harry Nyquist. Certain topics in telegraph transmission theory. *Transactions of the American Institute of Electrical Engineers*, 47(2):617–644, 1928.

[19] Giambattista Parascandolo, Heikki Huttunen, and Tuomas Virtanen. Taming the waves: sine as activation function in deep neural networks. 2016.

[20] 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.

[21] Hallison Paz, Daniel Perazzo, Tiago Novello, Guilherme Schardong, Luiz Schirmer, Vinicius da Silva, Daniel Yukimura, Fabio Chagas, Helio Lopes, and Luiz Velho. Mr-net: Multiresolution sinusoidal neural networks. *Computers & Graphics*, 2023.

[22] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pages 5301–5310. PMLR, 2019.

[23] Vishwanath Saragadam, Daniel LeJeune, Jasper Tan, Guha Balakrishnan, Ashok Veeraraghavan, and Richard G Baraniuk. Wire: Wavelet implicit neural representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18507–18516, 2023.

[24] Luiz Schirmer, Tiago Novello, Guilherme Schardong, Vinícius da Silva, Hélio Lopes, and Luiz Velho. How to train your (neural) dragon. In

Conference on Graphics, Patterns and Images, 36. (SIBGRAPI), 2023. URL `http://urlib.net/ibi/8JMKD3MGPEW34M/49T46US`.

[25] Jonathan Schwarz and Yee Whye Teh. Meta-learning sparse compression networks. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL `https://openreview.net/forum?id=Cct7kqbHK6`.

[26] Claude E Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, 1949.

[27] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020.

[28] Han Song, Mao Huizi, and Dally William, J. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR*, 2016.

[29] Ying Song, Jiaping Wang, Li-Yi Wei, and Wencheng Wang. Vector regression functions for texture compression. *ACM Transactions on Graphics (TOG)*, 35(1):1–10, 2015.

[30] Josep M Sopena, Enrique Romero, and Rene Alquezar. Neural networks with periodic and monotonic activation functions: a comparative study in classification problems. 1999.

[31] Yannick Strümpler, Janis Postels, Ren Yang, Luc Van Gool, and Federico Tombari. Implicit neural representations for image compression. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, pages 74–91, Cham, 2022. Springer Nature Switzerland. ISBN 978-3-031-19809-0.

[32] Jihoon Tack, Jongjin Park, Hankook Lee, Jaeho Lee, and Jinwoo Shin. Meta-learning with self-improving momentum target. *Advances in Neural Information Processing Systems*, 35:6318–6332, 2022.

[33] 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. *Advances in Neural Information Processing Systems*, 33:7537–7547, 2020.

[34] Ivana Tošić and Pascal Frossard. Dictionary learning. *IEEE Signal Processing Magazine*, 28(2):27–38, 2011.

[35] Zhijie Wu, Yuhe Jin, and Kwang Moo Yi. Neural fourier filter bank. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14153–14163, 2023.

[36] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. In *Computer Graphics Forum*, volume 41, pages 641–676. Wiley Online Library, 2022.

[37] Runzhao Yang, Tingxiong Xiao, Yuxiao Cheng, Qianni Cao, Jinyuan Qu, Jinli Suo, and Qionghai Dai. Sci: A spectrum concentrated implicit neural compression for biomedical data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 4774–4782, 2023.

[38] Wang Yifan, Lukas Rahmann, and Olga Sorkine-hornung. Geometry-consistent neural shape representation with implicit displacement fields. In *International Conference on Learning Representations*, 2021.

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

[40] Andreas Zell, Nuri Benbarka, Timon Hoefer, and Hamd Ul Moqueet Riaz. Seeing implicit neural representations as fourier series. In *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. IEEE Computer Society, 2022.