Computers & Graphics 114 (2023) 387–400

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Special Section on SIBGRAPI 2022

MR-Net: Multiresolution sinusoidal neural networks

Hallison Paz [a,*], Daniel Perazzo [a], Tiago Novello [a], Guilherme Schardong [a,c], Luiz Schirmer [a,c], Vinícius da Silva [b], Daniel Yukimura [a], Fabio Chagas [a], Hélio Lopes [b], Luiz Velho [a]

[a] *IMPA - VISGRAF Laboratory, Rio de Janeiro, Brazil*
[b] *PUC-Rio, Rio de Janeiro, Brazil*
[c] *Universidade de Coimbra, Coimbra, Portugal*

ARTICLE INFO

Article history:
Received 26 February 2023
Received in revised form 16 May 2023
Accepted 21 May 2023
Available online 26 May 2023

Keywords:
Multiresolution
Level of detail
Neural networks
Imaging

ABSTRACT

We present MR-Net, a general architecture for multiresolution sinusoidal neural networks, and a framework for imaging applications based on this architecture. We extend sinusoidal networks, and we build an infrastructure to train networks to represent signals in multiresolution. Our coordinate-based networks, namely L-Net, M-Net, and S-Net, are continuous both in space and in scale as they are composed of multiple stages that progressively add finer details. Currently, band-limited coordinate networks (BACON) are able to represent signals at multiscale by limiting their Fourier spectra. However, this approach introduces artifacts leading to an image with a ringing effect. We show that MR-Net can represent more faithfully what is expected of sequentially applying low-pass filters in a high-resolution image. Our experiments on the Kodak Dataset show that MR-Net can reach comparable Peak Signal-to-Noise Ratio (PSNR) to other architectures, on image reconstruction, while needing fewer additional parameters for multiresolution. Along with MR-Net, we detail our architecture's mathematical foundations and general ideas, and show examples of applications to texture magnification, minification, and antialiasing. Lastly, we compare our three MR-Net subclasses.

1. Introduction

In recent years, the computer science community has seen an explosion in research in neural networks, motivated mainly by advances in Deep Learning [1,2]. For visual computing, this was spurred by the creation of Convolutional Neural Networks (CNNs) [3], which had a significant impact both for the research community and the society at large [4,5]. The effectiveness of CNNs comes from the translation invariant properties of the convolution operator, which makes it a proper architecture for the analysis of visual imagery.

Deep neural networks such as CNNs, employ an array-based discrete representation of the underlying signal. In this case, the network input consists of a vector of pixel values (in RGB) representing the image *directly* by data samples. We call this kind of network a *data-based* network.

Moreover, the revolution in the media industry caused by deep neural networks motivated the development of new image representations using neural networks. While the data-based network is appropriate for analysis tasks, relying on a discretization of the image, another kind of network called *coordinate-based network* is suitable for synthesis, and provides a continuous and compact representation. For its characteristics, there is a growing interest in using these networks in imaging applications [6]. For instance, coordinate-based networks have been successfully applied in image compression [7] and super-resolution [8].

A coordinate-based network represents the image *indirectly* using a fully connected *multi-layer perceptron* (MLP) that takes as input a pixel coordinate and outputs a RGB color. These networks provide a continuous implicit representation for images [9], and allow for various applications, from Neural Signed Distance Functions (NeuralSDFs) [10] to Neural Radiance Fields (NeRFs) [11]. Since the coordinates are continuous, images can be presented in arbitrary resolution.

Imaging applications benefit greatly from multiresolution representations, as they allow us to represent an image hierarchically at different levels of details. This hierarchical model is aligned with some classical image and human visual perception models [12], and it is instrumental for many tasks in computer vision and graphics, such as compression, analysis and rendering.

* Corresponding author.
E-mail addresses: hallpaz@impa.br (H. Paz), daniel.perazzo@impa.br (D. Perazzo), tiago.novello@impa.br (T. Novello), guilherme.schardong@isr.uc.pt (G. Schardong), luiz.schirmer@isr.uc.pt (L. Schirmer), dsilva.vinicius@gmail.com (V. da Silva), yukimura@impa.br (D. Yukimura), fabio.chagas@impa.br (F. Chagas), lopes@inf.puc-rio.br (H. Lopes), lvelho@impa.br (L. Velho).

https://doi.org/10.1016/j.cag.2023.05.014
0097-8493/© 2023 Elsevier Ltd. All rights reserved.

For example, in image compression, we can use a multiresolution representation to identify and discard details that are not perceptually important, while preserving important features of the image. This can significantly reduce the amount of data needed to represent the image, while still maintaining its overall quality [13]. Additionally, in rendering, multiresolution representations have built-in support for antialiasing, which traditionally is implemented using image pyramids [14]. Another important applications of imaging in Graphics is texture synthesis. In that realm, besides antialiasing, the creation of visual patterns from examples has great relevance [15].

Traditionally, multiresolution representations for images have been based on signal processing techniques derived from Fourier theory [16]. Such operators were primarily motivated by image compression [17] and played an important role in the development of JPEG-2000 [18]. For instance, the Discrete Cosine Transform [19] have been used as an efficient way to get a frequency content of the image, and estimate the importance of certain coefficients to the image quality perception. Although Fourier transforms were initially popular for decomposing signals into multiple frequencies, the wavelet transform, introduced by Mallat [20], soon gained popularity as it enabled representing a signal in levels of detail and scale, and was widely applied to a first generation of wavelet-based image codecs [21]. Subsequently, wavelet analysis of multiscale edges led to a second generation of image coding methods, with higher compression rates [22]. Nevertheless, understanding and controlling the frequencies present in a signal has been critical on interpreting its details and avoiding artifacts such as aliasing. In this sense, sinusoidal functions have been instrumental in the development of the multiresolution theory.

Sinusoidal neural networks are examples of coordinate-based networks in which their activation function is the sine function. As such, they bridge the gap between the spatial and spectral domains, given the close relationship of the sine function with the Fourier basis. However, these sinusoidal neural networks have been regarded as difficult to train [23]. To overcome this problem, Sitzmann et al. [24] proposed a sinusoidal network for signal representation called SIREN. One of the key contributions of this work is the initialization scheme that guarantees stability and good convergence. Furthermore, it also allows modeling fine details in accordance with the signal's frequency content.

A *multiplicative filter network* (MFN) is a sinusoidal network simpler than SIREN which is equivalent to a shallow sinusoidal network [25]. Lindell et al. [26] presented *band-limited coordinate network* (BACON), an MFN that produces intermediate outputs with an analytical spectral bandwidth (specified at initialization) and achieves multiresolution of the underlying signal. While its structure allows BACON to be expressed as a linear combinations of sines, avoiding the composition of sines present in sinusoidal MLPs, it creates multiresolution representations by truncating the frequency spectra of the signals. This approach produces ringing artifacts in some levels of detail, and becomes evident when we look at the Fourier transform of the images.

The control of frequency bands in the representation is closely related with the capability of adaptive reconstruction of the signal in multiple levels of detail. In that context, Müller et al. [27] developed a multiresolution neural network architecture based on hash encoding. Also, Martel et al. [28] designed an adaptive coordinate network for neural signals.

In this context, we introduce *multiresolution sinusoidal neural networks* (MR-Net) based on classical signal multiresolution representations. Our results, presented in Section 5.2, indicate that using MR-Net produces better results compared to the previous state-of-the-art technique, BACON, while employing a smaller number of parameters. We describe three MR-Net subclasses: S-Net, L-Net and M-Net. Finally, we present applications on antialiasing and level-of-detail reconstruction.

In summary, we make the following contributions:

- We extend the sinusoidal neural networks and introduce a family of multiresolution coordinate-based networks, with unified architecture, that provides a continuous representation spatially and in scale. For this, we show how the initialization proposed for SIREN [24] can be better explored to control the frequencies learned by the model, and how we can decompose a network in multiple stages, inspired by the multiresolution analysis.
- We develop a framework for imaging applications based on this architecture, leveraging classical multiresolution concepts such as pyramids. Also, we show that this approach can more faithfully represent the frequency spectra of multiresolution signals, avoiding artifacts present in BACON [26] representation.
- We show that our architecture can represent images with good visual quality, being competitive with related methods in PSNR and number of parameters; we also demonstrate its use in applications of texture magnification and minification, and antialiasing.

This paper is an extension of our previous work [29], where we introduced a class of neural networks called *Multiresolution Neural Networks* (MR-Net). In this work, we present a complete description of this class of networks along with the motivations and a detailed mathematical model (see Section 2). We added Section 3, which explains our framework in detail and should help the construction of an independent implementation. We expanded the comparisons to include an analysis of the frequency spectra of the representations provided by MR-Net and BACON (see Section 5.2). We also present a broader set of experiments (see Section 5.3), using the Kodak dataset [30], for comparison of the MR-Net with other network architectures. Finally, we added Section 6, which compares the different MR-Net subclasses (S-Net, L-Net, and M-Net) in both qualitative and quantitative aspects.

2. Multiresolution sinusoidal neural networks

This section presents *multiresolution sinusoidal neural networks* (MR-Net) to represent signals in multiple levels of detail using sinusoidal MLPs.

2.1. Overview of the MR-Net framework

Our proposal is a family of coordinate-based networks with an unified architecture, and a framework for training neural networks on multiscale signal representation. We derive three main subclasses of MR-Nets: *S-Net*, *L-Net*, and *M-Net*. Each of them offers different trade-offs in terms of frequency control in the signal representation.

The characteristics of the MR-Net framework are:

- Flexible training data — the input signal can be given either by regular sampling, by multiresolution structure such as pyramids, or by stochastic/stratified sampling.
- 2 Types of Level of Detail — when training with multiresolution data, the level of details are determined by spectral projections, whereas with a single-resolution input signal, the level of details are determined by the network capacity.
- Progressive Training — the network is trained progressively, each stage at a time, using a variety of schedule regimes.
- Continuous Multiscale — the representation is continuous both in space and scale. Therefore, it can reconstruct the signal at any desired resolution/level of detail.

The next sections present the motivations and concepts underlying our approach, as well as the MR-Net architecture.

2.2. Motivation

Let $\mathscr{f} : \mathcal{D} \to \mathcal{C}$ be a *signal*, the *ground-truth*, where $\mathcal{D}$ and $\mathcal{C}$ are finite vector spaces representing the domain and codomain of $\mathscr{f}$. For instance, to represent an image, we can choose $\mathcal{D} = \mathbb{R}^2$ to represent the image's support and $\mathcal{C} = \mathbb{R}^3$ to represent the RGB *color space*. Throughout the text, the cursive letter will indicate the ground-truth while the standard letter indicates the corresponding neural network.

We can decompose the signal into a sum $\mathscr{f} = \mathscr{g}_0 + \cdots + \mathscr{g}_{N-1}$ of N stages, where $\mathscr{g}_0$ is its coarsest approximation and $\mathscr{g}_i$, for $i > 0$, progressively introduce finer details. The *level of detail i* is defined as $\mathscr{f}_i = \mathscr{g}_0 + \cdots + \mathscr{g}_i$, or as $\mathscr{f}_i = \mathscr{f} - (\mathscr{g}_{i+1} + \cdots + \mathscr{g}_{N-1})$. Thus, the stages can be defined as

$$\mathscr{g}_i = \mathscr{f}_{i+1} - K * \mathscr{f}_{i+1}, \text{ with } \mathscr{f}_{N-1} = \mathscr{f}.$$

That is, each stage $\mathscr{g}_i$ is the difference between the level $i+1$ and its convolution with a low-pass filter K. For example, K could be a *Gaussian kernel* $G(x, t) = \frac{1}{2\pi t} \exp\left(-\frac{\|x\|^2}{2t}\right)$, where t is the scale parameter. The sequences $\{\mathscr{f}_i\}$ and $\{\mathscr{g}_i\}$ resemble the *Gaussian* and *Laplacian pyramids*, which are widely used in the multiresolution analysis of digital images [31–33].

We address the problem of representing a signal $\mathscr{f}$ in multiresolution using sinusoidal neural networks. Motivated by the decomposition $\mathscr{f} = \mathscr{g}_0 + \cdots + \mathscr{g}_{N-1}$, we consider an aggregation of N sinusoidal MLPs $g_i : \mathcal{D} \to \mathcal{C}$, which we call stages, to approximate $\mathscr{f}$. Therefore, we propose training each network stage g_i by fitting it to the stages $\mathscr{g}_i$ of $\mathscr{f}$. This approach allows us to learn the frequency of $\mathscr{f}$ in a controlled manner, starting from lower details and gradually moving towards higher ones.

2.3. MR-Net architecture

We define the MR-Net as a function $f : \mathcal{D} \times [0, N] \to \mathcal{C}$, expressed as follows:

$$f(x, t) = c_0(t)g_0(x) + \cdots + c_{N-1}(t)g_{N-1}(x), \tag{1}$$

Each function $g_i : \mathcal{D} \to \mathcal{C}$ is a sinusoidal MLP (see Section 2.4 for its precise definition) and represents the ith *stage* of f, whose contribution is controlled by the function

$$c_i(t) = \max\Big\{0, \min\big\{1, t - i\big\}\Big\}. \tag{2}$$

Note that when $t < i$, $c_i(t) = 0$; when $i \le t \le i+1$, $c_i(t) = t - i$; and when $t > i+1$, $c_i(t) = 1$. Therefore, if $t = k + \delta$ with $k \in \mathbb{N}$ and $0 \le \delta \le 1$, we obtain

$$f(x, t) = g_0(x) + \cdots + g_k(x) + \delta g_{k+1}(x).$$

$f_t := f(\cdot, t) : \mathcal{D} \to \mathcal{C}$ is the *level of detail* t of the MR-Net f. These levels evolve continuously, allowing us to encode a continuous multiresolution of f at full resolution $f_N = g_0 + \cdots + g_{N-1}$. For this, we propose fitting f to the multiresolution given by the ground-truth signal $\mathscr{f} = \mathscr{g}_0 + \cdots + \mathscr{g}_{N-1}$. We initialize the parameters of each stage g_i with sufficient capacity to fit the stage $\mathscr{g}_i$ (see Section 2.6), then train each g_i to approximate $\mathscr{g}_i$.

The resulting MR-Net f learns the decomposition of the ground-truth signal $\mathscr{f}$ as a projection into the coarse scale space and a sequence of finer detail spaces. For instance, the initial stage $f_1 = g_0$ provides the least detailed approximation of f_N. The subsequent stages, represented by g_i with $i < 0$, progressively introduce finer details and are regulated by the scale parameter t. Essentially, the multiresolution can be navigated using the scale parameter t within the interval $[0, N]$, which makes this architecture closely aligned with the Multiresolution Analysis [20]. Fig. 1 shows the structure of a MR-Net having N stages.

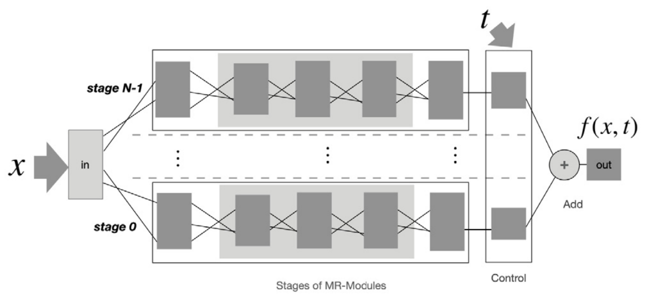

Fig. 1. Anatomy of the MR-Net family.

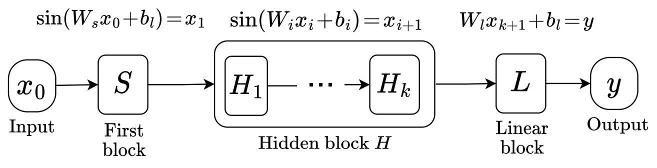

Fig. 2. General Anatomy of a MR-Module.

2.4. MR Module

Each stage g_i of the MR-Net f is a sinusoidal MLP, called *MR-Module*, that learns a signal representation as a combination of sinusoidal functions with induced frequency band.

We write each stage as $g_i = L_i \circ H_i \circ S_i$. The first layer projects the input x into a list of sines $S_i(x)$, which is the input of the composition H_i of the MLP hidden layers. The output $H_i \circ S_i(x)$ is a dictionary of sine combinations that are passed to the linear layer L_i. Finally, the control layer $c_i(t)$, with $t \in [i, i+1]$, blends the stage g_i with the level of detail $f(x, i-1)$.

There are two kinds of MR-Modules: *Pure sine* MR-Module ($H_i = \emptyset$) and *modulated sine* MR-Module ($H_i \ne \emptyset$). In the first case the network has only the first layer and the linear layer, and in the second case the network has the three blocks, including the hidden layers (see Fig. 2).

Without loss of generality, let us assume that the ground-truth signal is a grayscale image unless stated to the contrary. In this case, the stages of the MR-Net will have $\mathbb{R}^2$ and $\mathbb{R}$ as domain and codomain respectively. Consequently, a MR-Net with N stages $g_i : \mathbb{R}^2 \to \mathbb{R}$ has the form $f : \mathbb{R}^2 \times [0, N] \to \mathbb{R}$. Under this assumption, we present the building blocks of the MR-Net with details.

2.4.1. Pure sine MR-Module

The pure sine MR-Module is a shallow sinusoidal MLP defined as the composition $L \circ S$ of a sinusoidal layer S with a linear layer L. The layer $S : \mathbb{R}^2 \to \mathbb{R}^m$ projects the input x into a dictionary of m sines of the form $S(x) = \sin(W_s x + b_s)$, where $W_s \in \mathbb{R}^{m \times 2}$ and $b_s \in \mathbb{R}^m$ are the *weight matrix* and *bias*. The integer m is the *width* of the MR-Module. The layer $L : \mathbb{R}^m \to \mathbb{R}$ is an affine map $L(x) = W_l x + b_l$, where $W_l \in \mathbb{R}^{1 \times m}$ and $b_l \in \mathbb{R}$ are the final weight matrix and bias. Hence, $L \circ S$ acts as a spectral filter controlled by the initialization of W_s.

Since the risk landscape for the network loss has many local minima, it is expected that the weights W_s cannot move much further than their initialization. Therefore, W_s determine the range of frequencies one can filter from the input signal. Fig. 3 shows the structure of the $L \circ S$ and an example of a signal consisting of a linear combination of two frequencies. In this example, we have the layers $S : \mathbb{R} \to \mathbb{R}^2$, $S(x) = (sin(x), sin(5x))$ and $L : \mathbb{R}^2 \to \mathbb{R}$, $L(x_1, x_2) = x_1 + x_2$.

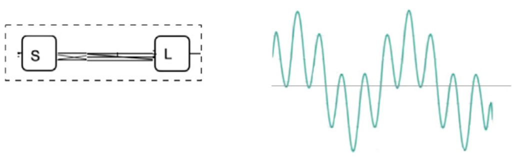

Fig. 3. Pure sine MR-Module.

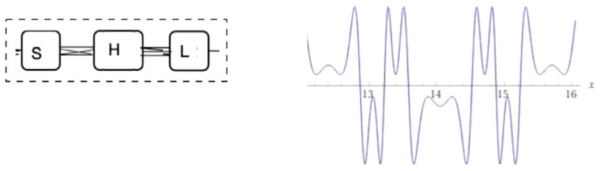

Fig. 4. Modulated sine MR-Module.

2.4.2. Modulated sine MR-Module

The Modulated MR-Module is a deep sinusoidal MLP $L \circ H \circ S$ that considers a hidden sinusoidal layer block $H : \mathbb{R}^m \to \mathbb{R}^m$. This map is the composition of k layers $H = H_k \circ \cdots \circ H_1$ with $H_i(x_i) = \sin(W_i x_i + b_i) = x_{i+1}$ where $x_0 \in \mathbb{R}^m$ is the input. The integers m and $k+1$ are the *width* and *depth* of the MR-Module.

The MR-Module $L \circ H \circ S$ is a linear combination of a dictionary of (spectral) atoms $H \circ S$ containing composition of sines. This fact has two consequences. First, the capacity of $L \circ H \circ S$ is greater than the capacity of $L \circ S$ with the same number of neurons [34]. Second, the initialization of $L \circ H \circ S$ does not directly control its frequency band as in the case of $L \circ S$.

In terms of localization properties in space and frequency, we can say the spectral atoms of the Modulated MR-Module are semi-local in the sense that these functions are adapted by the learning process to fit local variations of the signal (i.e., frequencies) across its (spatial) domain. This characteristic is evident in Fig. 4 that shows a network structure and the graph of $\sin\big(3\sin(5\sin(1.9x))\big)$. In this example, the first layer is represented by $S(x) = \sin(1.9x)$, the hidden block is the composition of two sine layers $H(x) = \sin\big(3\sin(5x)\big)$, and the linear layer is the identity function $L(x) = x$.

2.5. MR-Net subclasses

Here, we describe three subclasses of the MR-Net: *S-Net*, *L-Net*, and *M-Net*. The characteristics of their level of details depend on the configuration of their stages. Recall, from Eq. (1), that a MR-Net is a function $f : \mathbb{R}^2 \times [0, N] \to \mathbb{R}$, expressed as:

$$f(x, t) = c_0(t)g_0(x) + \cdots + c_{N-1}(t)g_{N-1}(x).$$

Therefore, to define the subclasses S-Net, L-Net, and M-Net, we only need to define the stages g_i.

2.5.1. S-Net

A S-Net is a MR-Net in which each stage g_i is a pure sine MR-Module, i.e. $g_i = L_i \circ S_i$ where S_i and L_i are the first and the linear layers (see Fig. 5). For this reason, each stage g_i can be expressed as a sum of sine functions

$$g_i(x) = a_0 + \sum_{j=1}^{m} a_j \sin\left(\omega_j x + \varphi_j\right),$$

where the *frequencies* ω_j and *phase-shifts* φ_j are given by the weights and bias of the first layer S_i. The *amplitudes* a_j are given by the linear layer L_i. As a consequence, the resulting S-Net f can provide level of detail based on the controlled initialization of frequency band of each stage g_i (see Section 2.6).

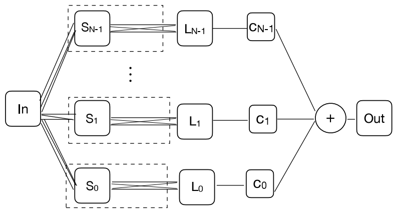

Fig. 5. S-Net architecture.

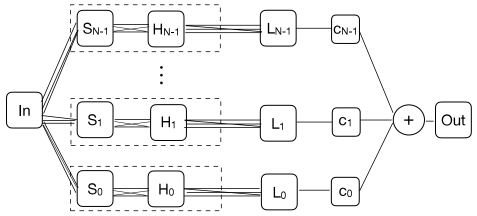

Fig. 6. L-Net architecture. Note that the stages are independent of each other.

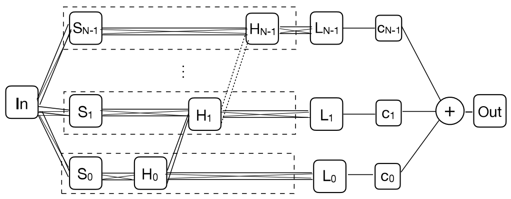

Fig. 7. M-Net architecture. Notice how the hidden-layers blocks of adjacent MR-Modules are connected.

2.5.2. L-Net

A L-Net is a MR-Net f in which each stage $g_i = L_i \circ H_i \circ S_i$ is an independent modulated sine MR-Module: S_i, H_i, L_i are the first, hidden, and linear blocks (see Fig. 6). For this reason, the level of detail capacity of f is determined by the width and depth of each stage g_i.

The Nth level $f(\cdot, N)$ of a L-Net is an example of a sinusoidal MLP. Without loss of generality, let $f(\cdot, t) = c_0(t)g_0 + c_1(t)g_1$ be a L-Net with two stages having the same architecture with a single hidden layer. We show that $f(\cdot, 2) = L \circ H \circ S$. For this, define the matrices of S, H, and L, respectively, using $W_s = \begin{pmatrix} W_s^0 \\ W_s^1 \end{pmatrix}$, $W_h = \begin{pmatrix} W_h^0 & 0 \\ 0 & W_h^1 \end{pmatrix}$, $W_l = \begin{pmatrix} W_l^0 & W_l^1 \end{pmatrix}$, where W_s^j, W_h^j, W_l^j are the matrices of the stages g_j. The biases are defined in a similar way. Such procedure can be extended to a sum of N stages in a analogous way. Thus, the L-Net gives us a controllable way of increasing the *width* of a sinusoidal MLP during training.

2.5.3. M-Net

While the stages of S-Nets and L-Nets are independent, with M-Net we propose a way to reuse information learned at previous stages. A M-Net is a MR-Net where each stage g_i is a modulated MR-Module linked to its subsequent stage g_{i+1}. That is, $H_i \circ S_i$ is composed both with the linear layer L_i, producing the stage $g_i = L_i \circ H_i \circ S_i$, and the hidden block H_{i+1} resulting in $g_{i+1} = L_{i+1} \circ H_{i+1} \circ (S_{i+1}, H_i \circ S_i)$ (Fig. 7). This way, the hidden block has the form $H_{i+1} : \mathbb{R}^{2m} \to \mathbb{R}^m$; m is the number of neurons.

The hidden block of a stage of the M-Net is composed with a sequence of hidden blocks coming from previous stages. That is, the M-Net contains a deep MLP $L_{N-1} \circ \overline{H_{N-1}} \circ \cdots \circ \overline{H_1} \circ H_0 \circ S_0$ with N hidden blocks; $\overline{H_i}$ is the part of H_i connecting to the stage g_{i+1}. Thus, the M-Net also allows us to increase the *depth* of a

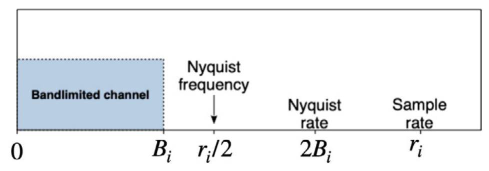

Fig. 8. Nyquist limit.

sinusoidal MLP during training implying that the capacity of each stage increases with its depth in the hierarchy. This feature makes the M-Net a suitable and compact architecture for multiresolution training, as discussed in Section 6, and we will utilize this subclass in the applications of Section 4.

2.6. Frequency control

Let $\mathscr{f} = \mathscr{g}_0 + \cdots + \mathscr{g}_{N-1}$ be the ground-truth signal in multiresolution, and $f : \mathbb{R}^2 \times [0, N] \to \mathbb{R}$ be a MR-Net with N stages. To train each stage g_i, we propose to initiate its parameters based on the frequency content of $\mathscr{g}_i$. There are two ways to control the frequency band of g_i. First, we can initialize the weight matrix of the first layer of g_i. Second, we can vary its width and depth. Moreover, these mechanisms can be combined.

2.6.1. Frequency initialization

Training sinusoidal MLPs can be challenging, as periodic activation functions may lead to instability in deep architectures [23]. Sitzmann et al. [24] propose an initialization scheme that guarantee stability during training. They initialize the weights $W = \omega\,\tilde{W}$ of the MLP first layer $\sin(Wx + b)$ such that $\tilde{W}$ is uniformly sampled in $[-1, 1]$ and the number ω controls the range of frequencies. That is, the layer projects the input x in a list of sines with frequencies in $[-\omega, \omega]$. They empirically choose $\omega = 30$ in their experiments. For the hidden layers, they choose the weights distributed uniformly in $\left(-\sqrt{6/m}, \sqrt{6/m}\right)$, where m is the width of the layer. See [24] for the details.

Regarding the initialization of the MR-Net f, let $g_i = L_i \circ H_i \circ S_i$ be its ith stage, where S_i, H_i, and L_i are its first, hidden, and linear blocks. Observe that each coordinate of the sinusoidal layer $S_i(x) = \sin\left(W_{s_i}x + b_{s_i}\right)$ has the form $\sin(\omega_1 x_1 + \omega_2 x_2 + \varphi)$, where the *frequencies* ω_1 and ω_2 form a line of the matrix W_{s_i}, $x = (x_1, x_2)$ is the input, and the *phase-shift* φ is a coordinate of the bias b_{s_i}. We follow the above initialization approach to initialize g_i. However, instead of using $\omega = 30$ we consider it to be a *bandlimit frequency* on the ground-truth stage $\mathscr{g}_i$.

Ideally, the initialization of frequencies in g_i should match the frequency content of the ground-truth signal at the stage $\mathscr{g}_i$. However, as we usually do not have access to this information, we opt to use an upper bound. Specifically, we assume that $\mathscr{g}_i$ has no frequency higher than a *bandlimit* B_i. The Nyquist–Shannon sampling theorem says that we can reconstruct $\mathscr{g}_i$ from a sample $\{\mathscr{g}_i(x_{kl})\}$, where the regular grid x_{kl} has points spaced with size $\frac{1}{r_i} < \frac{1}{2B_i}$. The number r_i is the *sample rate*. Fig. 8 illustrates such requirement using the frequency rate notation.

Therefore, to fit the stage g_i to the sample $\{\mathscr{g}_i(x_{kl})\}$, we opt to initialize g_i such that it can represent frequencies up to $\omega_i := \frac{r_i}{2}$. For this, we initialize the lines of the first matrix W_{s_i} of g_i uniformly in the set $\Omega_i = [-\omega_i, \omega_i]^2$. Thus, the sinusoidal layer S_i may contain frequencies already initialized at previous stages because $\{\mathscr{g}_i\}$ is a Laplacian pyramid of $\mathscr{f}$, thus, $\Omega_0 \subset \cdots \subset \Omega_{N-1}$. The weights of the hidden block H_i are initialized following the same scheme in [24].

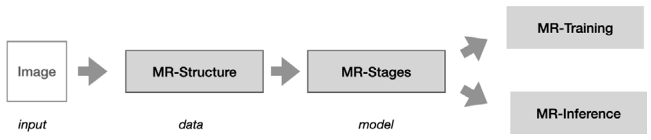

Fig. 9. MR-Net framework components.

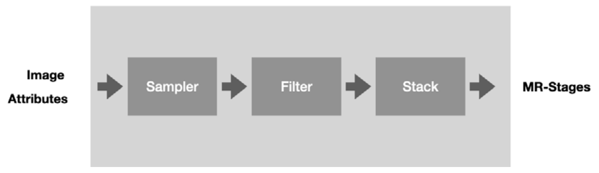

Fig. 10. MR-Structure.

2.6.2. MR-Module capacity

The capacity of each MR-Net stage $g_i = L_i \circ H_i \circ S_i$ is controlled by its width m and depth $k + 1$. The width m determines that an input x will be transformed into a list of m sines $S_i(x) = \sin\left(W_{s_i}x + b_{s_i}\right)$, where W_{s_i} has rows sampled at the set Ω_i defined in Section 2.6.1. By increasing m, the dictionary of input frequencies is augmented. The hidden block H_i consisting of k sinusoidal layers, further enhances this list of frequencies.

Regarding the training of the MR-Net stage g_i, we recall that it is a MLP. Rahaman et al. [35] shows that during training a MLP learns lower frequencies first, a phenomenon known as *spectral bias*. They also show that increasing the network depth (for fixed width) improves the network's ability to fit higher frequencies, while increasing the width (for fixed depth) also helps, but the effect is considerably weaker.

To mitigate the effects of the spectral bias, we employ the above frequency initialization approach to the stage g_i. This ensures that the training of g_i begins with frequencies that are *close* to those present in the corresponding ground-truth data $\mathscr{g}_i$.

Furthermore, the MR-Net architecture enables us to gradually increase its width and depth by adding stages (see Sections 2.5.2 and 2.5.3), providing us with a controlled way of increasing network capacity while allocating ground-truth frequencies across stages in a controllable manner. For example, a L-Net can be viewed as a specific MLP where adding a new stage increases its width. As a result, L-Net allows us to divide a given MLP into stages. On the other hand, M-Net has a more sophisticated structure, as adding a stage results in a network that is wider and deeper.

3. MR-Net in detail

This section presents the MR-Net framework in detail by conceptually dividing it in four main components: *MR-Structure*, *MR-Stages*, *MR-Training*, and *MR-Inference* (see Fig. 9).

Let $\mathscr{f} : \mathcal{D} \to \mathcal{C}$ be the ground-truth signal (*input data*), and $f : \mathcal{D} \times [0, N] \to \mathcal{C}$ be a MR-Net with N stages $\{g_i\}$ to fit $\mathscr{f}$ in multiresolution. We use this setting to present the framework.

3.1. MR-Structure

The MR-Structure is a data structure that encapsulates the input data $\mathscr{f}$. It includes $\mathscr{f}$ and metadata about its *sampling mode*, *filtering type*, and the *multi-stage stack* (see Fig. 10).

The training of MR-Net stage g_i receives pairs $\{x_j, y_j\}$ as input, with points x_j sampled in the domain $\mathcal{D}$ of $\mathscr{f}$ and $y_j = \mathscr{f}(x_j) \in \mathcal{C}$. For a squared image, we could have $\mathcal{D} = [-1, 1]^2 \subset \mathbb{R}^2$. During training, we consider the signal $\mathscr{f}$ to have codomain in

Fig. 11. Image values (RGB and monochromatic).

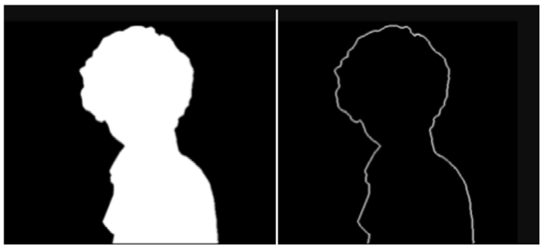

Fig. 12. Examples of image attributes (mask and edges).

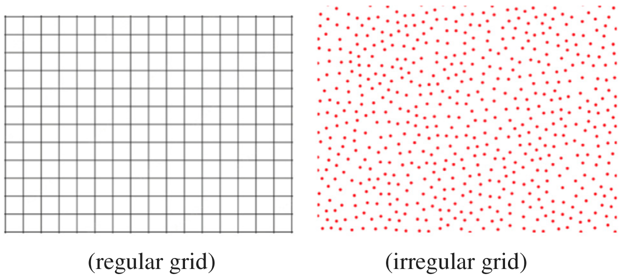

Fig. 13. Sampling modes.

monochromatic ($\mathcal{C} = \mathbb{R}$) or RGB color ($\mathcal{C} = \mathbb{R}^3$) (see Fig. 11). Depending on the application, other color spaces or additional attributes, such as masks and features, could be used (see Fig. 12).

The data input for the training of each of the N stages g_i is organized in a *multi-stage stack* $\{x_j, y_j\}_i$. Specifically, for each i, the set of pairs $\{x_j, y_j\}_i$ is a sample of the ith level of detail f_i of the signal f or a sample of the original signal f, with $\{x_j\}_i \subset \mathcal{D}$ and $\{y_j\}_i = \{f_i(x_j)\}$. From the point of view of *representation theory*, we can interpret this as a projection of the function f_i onto the primal *Shannon basis* (i.e., Dirac delta distribution). In the context of signal processing, this basis is a sampling grid of impulses and the representation consists of the sequence $\{y_j\}_i$ at the grid locations $\{x_j\}_i$.

The type of sampling used to extract the multi-stage stack $\{x_j, y_j\}_i$ from the signal f is an important aspect in the MR-Net training. In that respect, it is instrumental to consider two types of samplings: *regular* and *irregular* (Fig. 13). In the regular case, the sampled points are organized in a regular grid discretization $\{x_{k,l}\}$ of the domain $\mathcal{D}$. For flexibility, we implement a sampler module that take regular samples or stochastic samples using the Poisson disk sampling [36,37].

The multi-stage stack of the input signal f could have different resolutions. For the regular case where the sampled points are organized in a regular grid $\{x_{k,l}\}$ (the highest resolution), we can structure the multi-stages in dyadic lattice following the 2^i rule:

$$\{x_{k,l}\}_i = \{x_{2k,2l}\}_{i+1} \text{ with } \{x_{k,l}\}_N = \{x_{k,l}\}.$$

Here we are assuming that $\{x_{k,l}\}$ is a grid of size $2^k \times 2^k$ for some integer $k > N$. Thus each dimension of a stage grid $\{x_{k,l}\}_i$ has twice the size of the previous one (Fig. 14(a)). On the other hand, we could consider that the sampling grids $\{x_{k,l}\}_i$ have a fixed resolution, and sample the signal f on its highest level or consider the level of detail f_i for each grid stage $\{x_{k,l}\}_i$ (Fig. 14(b)). See Section 3.3.4 for more details.

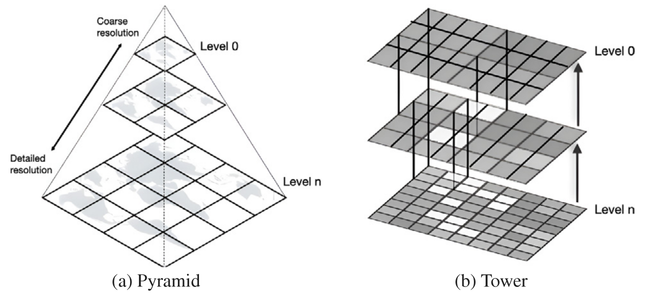

Fig. 14. Examples of multi-stage stacks.

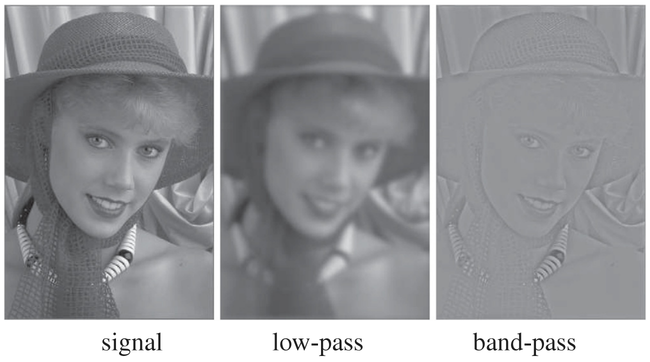

Fig. 15. Examples of filter types.

Each level i of the multi-stage stack can be filtered to separate the level of detail f_{i+1} into different frequency bands. In this sense, we can use the unfiltered signal f, a low-pass version of f_{i+1} or a band-pass version of f_{i+1}. For this, it is common to employ a Gaussian kernel as the low-pass kernel and a difference of Gaussians as the band-pass kernel (see Fig. 15). Precisely, we can filter f_{i+1} by convolving it with a Gaussian kernel K:

$$f_i(k, l) = (K * f_{i+1})(k, l) \tag{3}$$

We abuse the notation and denote by $f_i(k, l)$ the function f_i evaluated at $x_{k,l}$. Similarly, the ith band-pass stage $g_i = f_i - f_{i-1}$ is defined using $g_i(k, l) = (f_i - K * f_i)(k, l)$.

3.2. MR-Stages

The MR-Stages $\{g_i\}$ are the build blocks of the MR-Net f. They constitute a stack of MR-Modules that are interconnected according to the chosen MR-Net subclass as illustrated on Figs. 5, 6, and 7.

The MR-Net configuration is given by the parameters of the MR-Stages: number of stages, depth and width of the stages.

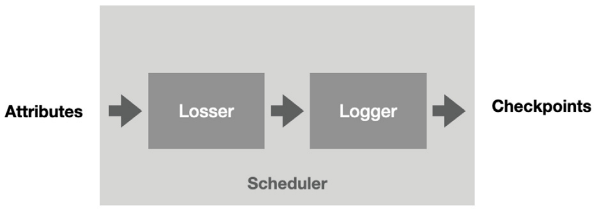

Fig. 16. MR-Training.

3.3. MR-Training

The training of the MR-Net f must consider the mechanisms for learning different levels of detail at each stage g_i of f. The capacity of g_i to represent the underlying ground-truth stage $\mathscr{g}_i$ can be achieved by initializing its first layer or adjusting its width and depth, as explained in Section 2.6. Regarding the network input, we can either use the original signal, or pre-process the signal with a low-pass filter.

The MR-Training incorporates a *loss functional*, a *logger* to monitor the training, as well as, a *scheduler* (see Fig. 16). We describe each of these components, as well as the multiresolution regime in the following sections.

3.3.1. Loss functional

Signal reconstruction is related to interpolation and fitting. We would like to fit the MR-Net f to the ground-truth signal $\mathscr{f}$ using a given sample of $\mathscr{f}$. For this, we observe that training f is related to a *regression* in multi-scale. This means that: i) the approximation should take into account a multiresolution $\mathscr{f} = \mathscr{g}_0 + \cdots + \mathscr{g}_{N-1}$ of the ground-truth signal; and ii) each MR-Net stage g_i should fit to a sample of $\mathscr{g}_i$.

To accomplish these goals above, we define a *loss functional* $\mathcal{L}_i$ to train each stage g_i of the MR-Net f. For this, we assume that the signal $\mathscr{f}$ was sampled and organized in a *multi-stage stack* $\{x_j, y_j\}_i$ such that $y_j = \mathscr{g}_i(x_j)$. Thus $\mathcal{L}_i$ is defined by minimizing the differences between the true values y_j at the sample points x_j and the predicted values $g(x_j)$ at the ith stage:

$$\mathcal{L}_i(\theta_i) = \frac{1}{K_i}\sum \left\|g_i(x_j) - y_j\right\|^2. \tag{4}$$

Where θ_i are the parameters of g_i and K_i is the size of $\{x_j, y_j\}_i$.

When the multi-stage stack $\{x_j, y_j\}_i$ is constructed by filtering $\mathscr{f}$ using a Gaussian filter (Eq. (3)), we should replace $\left\|g_i(x_j) - y_j\right\|$ by $\left\|f_i(x_j) - y_j\right\|$ in Eq. (4). Recall that f_i is the ith level of detail of the MR-Net f, i.e. $f_i = g_0 + \cdots + g_i$.

During training, the network can be over-fitted to the data. To avoid this, we explore regularization strategies such as defining convergence criteria for early stopping to fit the network to the data. In the future, we plan to enhance these strategies by adding *regularization terms* based on network derivatives.

3.3.2. Logger

The training of the MR-Net f is monitored by a logging module that helps to visualize the learning progress. During training, the logger receives messages when the network training starts and ends, when a stage training starts and ends, when a epoch training ends, and when a batch training ends.

This architecture allows the logger to be customized to accommodate different MR-Net configurations and different actions for visualizing the training progress. For instance, it is possible to write a logger to save partial results to the disk, display them in a development environment, or send them to a cloud based service.

3.3.3. Scheduler

Since a MR-Net f has N stages g_i, each learning a level of detail of the ground-truth signal $\mathscr{f}$, one important aspect is their training schedule. If the input multi-stage stack $\{x_j, y_j\}_i$ is organized as a Laplacian pyramid and f is a L-Net, it is possible to train all stages in parallel by summing the loss functions $\mathcal{L}_i$ of the MR-stages. However, in general, it may be beneficial to train each stage sequentially, from the lowest to the highest level. This scheduling is our choice and a common strategy in the traditional multiresolution analysis of signals.

Furthermore, we adopt a progressive learning strategy by "freezing" the weights of a stage once it is trained in the scheduled sequence. This strategy guarantees that the details are added to the representation incrementally from coarse to fine.

We also employ an adaptive training scheme for each stage optimization, combining both accuracy loss thresholds and convergence rates. The training process is outlined in Algorithm 1.

Algorithm 1: MR-Net training.

```
Data: A multi-stage stack $\{x_j, y_j\}_i$ with $N$ levels.
Result: A MR-Net $f$ with $N$ stages $g_i$.
1  Initialize a MR-Net model with a single stage $g_0$;
2  Notify Logger that network training will start;
3  for $stage \leftarrow 0$ to $N-1$ do
4      if $stage \neq 0$ then
5          Create a new stage $g_{stage}$ and add it to the model;
6          Freeze the parameters of the stage $g_{stage-1}$;
7      Notify Logger that stage training will start;
8      current_traindata ← multires_stack[$stage$];
9      for $epoch \leftarrow 0$ to current_limit_of_epochs do
10         for $batch$ in current_traindata do
11             Train $g_{stage}$ using the loss $\mathcal{L}_{stage}$ (Eq. (4)) ;
12             Notify Logger that batch training has finished;
13         Notify Logger that epoch training has finished;
14         if convergence_criteria_reached() then
15             break;
16     Notify Logger that stage training has finished;
17 Notify Logger that network training has finished;
```

3.3.4. Level of detail schemes

By combining the different aspects discussed in the previous sections we can define various schemes for learning level of detail representations using MR-Nets. The main ones are: capacity based with original signal; filtering with Gaussian/Laplacian tower; and filtering with Gaussian/Laplacian pyramid.

Capacity based with original signal. In this scheme, we train each stage g_i of the MR-Net f on the same sampling of the ground-truth signal $\mathscr{f}$ which we consider to have no frequency higher than a bandlimit ω. That is, the input multi-stage stack $\{x_j, y_j\}_i$ is composed of N copies of a sample $\{x_j, y_j\}$ of $\mathscr{f}$. Thus, the training of each stage g_i receives the same input. This scheme is based on the fact that even when f does not have enough capacity to fit $\mathscr{f}$, it can learn its lowest frequencies and represent a filtered version of $\mathscr{f}$. Section 2.6.2 describes the frequency control by network capacity, and Fig. 17 illustrates this phenomenon by showing the reconstruction in 1D example. The MR-Net used in this case has a single stage with width 16 and a single layer in all blocks.

To initialize the N stages g_i of f, we follow the scheme presented in Section 2.6.1. We initialize the rows of the first matrix of each stage g_i with values in $\Omega_i = [-\omega_i, \omega_i]^2$, where ω_i is a partition of the interval $[0, \omega]$, and ω is the bandlimit of $\mathscr{f}$. Consequently, g_0 is initialized and trained to learn the lowest frequencies of $\mathscr{f}$ up to a limit determined by its capacity. We

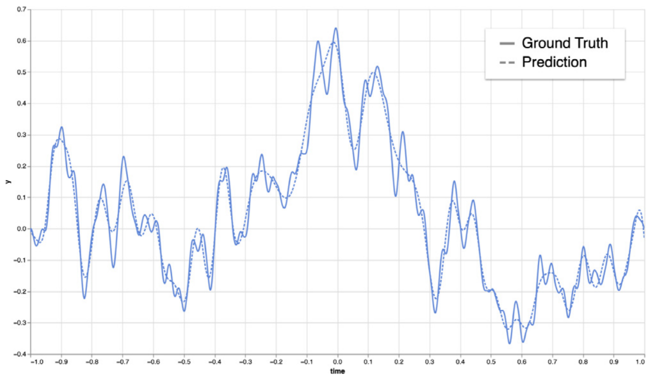

Fig. 17. Capacity-filtering in the reconstruction of a 1D signal.

then add stage g_1 to the network to learn more details of $\mathscr{f}$, and continue adding stages until the Nth stage is reached.

Note that, in this scheme, the training data and the test data for each level of detail is the original signal. This way, we can also stop adding stages if a desired error tolerance is achieved.

Filtering with Gaussian/Laplacian tower. To have control over the frequencies present in the signal $\mathscr{f}$, we sample a filtered multi-stage stack $\{x_j, y_j\}_i$ and train the stages g_i of MR-Net f to approximate each level i of this stack.

We start by feeding our model with a *Gaussian tower*, that is, a multi-stage stack $\{x_j, y_j\}_i$ where each level i is a version of the level $i+1$ filtered by a low-pass filter. Precisely, the Gaussian tower $\{x_j, y_j\}_i$ is defined recursively by convolving the *level of detail* $\mathscr{f}_i$ of $\mathscr{f}$ with a Gaussian kernel K:

$$\mathscr{f}_i(x_j) = (K * \mathscr{f}_{i+1})(x_j),$$
$$\mathscr{f}_{N-1} = \mathscr{f}.$$

Thus, we define $y_j = \mathscr{f}_i(x_j)$. This way, each level i is reconstructed with the same amount of samples. The MR-Net f must be trained from the less detailed scale to the most detailed one using the loss $\mathcal{L}_i$ that minimizes the differences $\left\|f_i(x_j) - y_j\right\|^2$, where $f_i = g_0 + \cdots + g_i$.

Similarly, we could represent the multi-stage stack $\{x_j, y_j\}_i$ as a *Laplacian tower* $\{\mathscr{g}_i\}$ by using:

$$\mathscr{g}_i(x_j) = (\mathscr{f}_i - K * \mathscr{f}_i)(x_j),$$
$$\mathscr{g}_0(x_j) = (K * \mathscr{f}_1)(x_j).$$

Thus, we define $y_j = \mathscr{g}_i(x_j)$ and train the MR-Net f using the loss $\mathcal{L}_i$ that minimizes the differences $\left\|g_i(x_j) - y_j\right\|^2$.

Filtering with Gaussian/Laplacian pyramid. The *Gaussian pyramid* is a classical multiscale representation of uniformly sampled signals. Based on the Shannon sampling theorem, the Gaussian tower is a highly redundant multiscale representation. On the other hand, the Gaussian pyramid is "critically sampled", i.e., it has the minimum number of samples required to represent each frequency band.

Precisely, the Gaussian pyramid $\{x_{k,l}, y_{k,l}\}_i$ is defined by recursively downsampling the above Gaussian tower of the signal $\mathscr{f}$ by a factor 2. As in Section 3.1, we are assuming that the sampled points $\{x_{k,l}\}$ forms a grid of size $2^k \times 2^k$ for some integer $k > N$. Similarly, the Laplacian pyramid is defined using the Laplacian tower of $\mathscr{f}$.

While the reconstruction of signals using the Gaussian tower is perfect, it is also wasteful if we can generalize correctly the model based only on the samples of a Gaussian pyramid. In terms of efficiency, it is faster to train the model on fewer samples, aligned with classical sampling theory results.

When training with a multi-stage stack with grids of different resolutions such as a Gaussian pyramid, we can build another multi-stage stack where each level has the same resolution as the original signal and use it as test data. In this case, this second multi-stage stack, a tower, should have each level filtered accordingly to its corresponding level in the pyramid, so that they are separated in similar frequency bands. With this pre-processing, we can train the network on the multiresolution pyramid data, and evaluate it on the original signal resolution, comparing it against the multiresolution tower data to check if it is generalizing as expected.

The training of the MR-Net stages g_i using Gaussian/Laplacian is analogous to the tower's case. Regarding the initialization of g_i, observe that $2^{k-N+1+i}$ is the height and width of the ith stack $\{x_{k,l}, y_{k,l}\}_i$, thus it cannot contain frequencies higher than $\omega_i = 2^{k-N+i}$. Thus we propose to initialize the frequencies of $\{g_i\}$ following a dyadic sequence of frequency bands $\omega_{N-1}, \omega_{N-2}, \ldots, \omega_0$, which is equivalent to

$$\omega_{N-1}, \frac{\omega_{N-1}}{2}, \ldots, \frac{\omega_{N-1}}{2^{N-1}} \text{ with } \omega_{N-1} = 2^{k-1}.$$

These are the bandlimits used to define the sets $\Omega_i = [-\omega_i, \omega_i]^2$ to initialize the frequencies of the first layer of each stage g_i.

3.4. MR-Inference

Arguably, the primordial purpose of a signal representation is to provide an accurate reconstruction of the underlying data. Moreover, in the ideal case, the reconstruction method should be able to work with a continuous model of the signal, generating signal values at arbitrary points of its domain.

In that respect, coordinate-based neural networks features a compact model of the signal as a continuous function. Additionally, our MR-Net architecture gives a representation that is continuous both at space and scale. Therefore, it can reconstruct the signal zooming in and out at any desired level of detail by specifying a value t to adjust the control layer coefficients, during the inference, according to Eqs. (1) and (2).

These characteristics are very important in media applications. In particular, there is a need to control the signal reconstruction for rendering, thus making it adapted to display resolution.

3.4.1. Antialiasing and progressive processing

The MR-Net architecture subsumes a model that incorporates filtering of the signal's frequency content in a controlled manner. This capability is crucial for antialiasing, necessary to avoid visual artifacts when rendering the signal.

The MR-Net representation as a hierarchy of levels of detail has implications for transmission and processing of the signal. On one hand, regarding the former, it is possible to send coarse versions of the signal, quickly through a channel and subsequently update the level of detail for progressive renderings. On the other hand, concerning the latter, level of detail facilitates data caching using the different memory structures of the GPU.

4. Imaging applications

This section describes the implementation and experiments of the MR-Net for imaging applications. We adopt the M-Net subclass of the architecture, and the level of detail scheme based on the Gaussian Pyramid in all examples. The multiresolution for the image Pyramid is according to a dyadic structure, i.e., 2^j. The Image Pyramid is built by filtering with a Gaussian kernel and decimation.

We have designed the MR-Module considering an empirical exploration of the sub-network capacity to represent images with

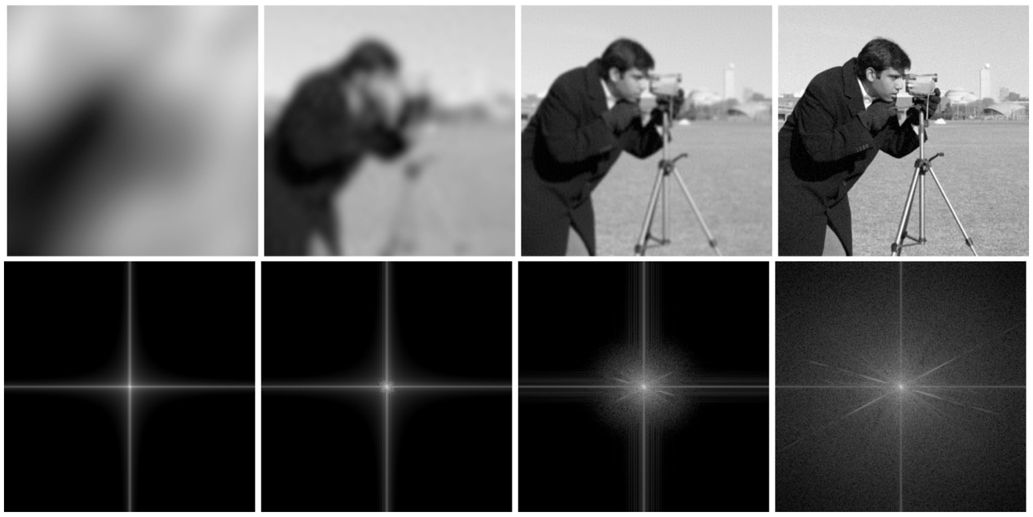
Fig. 18. Cameraman - reconstructed multiresolution levels 1, 3, 5 and 7 and corresponding Fourier spectra.

typical characteristics (i.e., photographs). Each stage has a width of 96 neurons for all layers on its first, hidden, and linear blocks. In each stage, the hidden block has only a single layer.

We have determined that the base resolution of the train data at the first stage should be $2^3 = 8$, and we have chosen the number of stages of the network based on the resolution of the image to be represented. Unless otherwise indicated, the images used in the experiments have a resolution of 512, in which case the pyramid is composed of the following resolution levels: 8, 16, 32, 64, 128, 256, and 512. Consequently, the network has a total of 7 stages.

We train the network using an adaptive scheme with the following hyper-parameters: Loss Function used is MSE (Mean Squared Error); convergence threshold = 0.001 (i.e., training of a stage stops if loss value changes less than 0.001%); maximum number of epochs per stage of 300; each epoch visits all the pixels once; size of mini-batch of 65 536 (to fit the GPU memory). Training uses Adam with a learning rate of $1e{-}4$.

The network initialization follows the scheme in Section 2.6.1, with the choice of frequency bands described in the “Filtering with Gaussian pyramid” level of detail scheme (Section 3.3.4).

4.1. Level of detail example

We now show an example of a multiresolution image representation using the setup described above. For this experiment, we chose the “Cameraman”, a standard test image used in the field of image processing and also in Sitzmann et al. [24]. The source is a monochromatic picture with 512 × 512 pixels of resolution.

Fig. 18 depicts the levels 1, 3, 5 and 7 of the multiresolution reconstructed with resolution of 512 × 512, as well as the corresponding Fourier spectra.

The training times for each stage of the networks are as follows: 5 s, 4 s, 3 s, 11 s, 17 s, 29 s and 48 s. The total training time is 117 s. The machine was a Windows 10 laptop with a NVIDIA RTX A5000 Laptop GPU. Note that these times result from the adaptive training regime and the number of samples for each level of the Gaussian Pyramid.

The training evolution is depicted in the graph of Fig. 19 that shows the convergence of the MSE loss with the number of epochs for each multiresolution levels 1, 3, 5, and 7. It is worth pointing out the qualitative behavior of the network, in that the base level (stage 1) takes more than 200 epochs to reach the limit, while detail levels (stages 3, 5, 7) take less than 150 epochs to

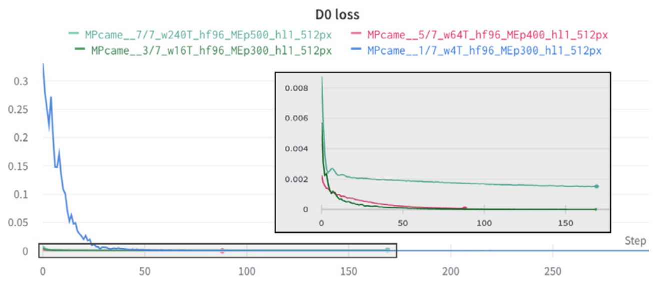

Fig. 19. Qualitative convergence behavior for Cameraman in Fig. 18.

converge. Also, the error decreases for each level of detail. It is like, there are two different modes, one to fit the base level and the other for the detail levels.

The inference time for image reconstruction, varies from 0.02 s on the GPU to 0.7 s on the CPU, which is sufficiently fast for interactive visualization.

4.2. Texture

The second imaging example is the usage of the MR-Net representation to model texture and patterns. Arguably, visual textures constitute one of the most important applications for images in diverse fields, ranging from photo-realistic simulations to interactive games. Currently, more and more image rendering relies on some kind of graphics acceleration, sometimes through GPUs integrated with Neural Engines. In that context, it is desirable to have a neural image representation that is compact and supports the level of detail.

For the experiment shown in this subsection we have chosen an image of woven fabric background with patterns. The characteristics of this texture allows us to explore the limits of visual patterns at different resolutions. The original image has a resolution of 1025 × 1025 pixels and the corresponding model contains 5 levels of detail.

In Fig. 20 we show our experiments, where Fig. 20(b) shows the image in the original resolution, while Fig. 20(a) shows a zoom-in and Fig. 20(c) shows a zoom-out.

The zoom-in is a detail with a resolution of 562 × 562. marked by the white rectangle in Fig. 20(b) and scaled up to 1025 × 1025. Thus a zoom-in factor of 1.8 times. It can be seen that the

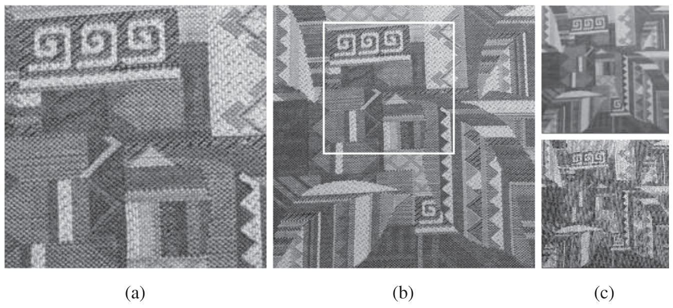
Fig. 20. Woven fabric texture: (a) zoom in; (b) original image resolution; (c) zoom out.

enlargement extrapolates the fine details of the image at this higher resolution beyond the original image.

The zoom-out is a reduction of the entire image to 118 × 118 pixels (shown enlarged to 501 × 501 in the image for better viewing). The top sub-image is the network reconstruction at the appropriate level of detail (approximately 0.92). The bottom sub-image is point-sample nearest neighbor reduction. It can be seen that our reconstruction is a proper anti-aliased rendition of the image, while the sampled reduction exhibits aliasing artifacts.

These two behaviors in the experiment are manifestations of "magnification" and "minification", classical resampling regimes for respectively scaling up and down the image [38]. In the first case, it is necessary to interpolate the pixel values, and in the second case, it is required to integrate pixel values corresponding to the reconstructed pixel. The M-Net model accomplishes these tasks automatically. Note that we have chosen a fractional scaling factors in both cases to demonstrate the continuous properties in space and scale of the M-Net model.

4.3. Anti-aliasing

In the previous subsection we resorted to the level of detail control to guarantee an alias free rendering independently of the sampling resolution. However, this task was facilitated because we could use a constant level of detail for the entire image, due to the zooming in/out operation in 2D.

On the other hand, in texture mapping applications, this scenario is no longer the case. Typically, it requires to map a 2D texture onto a 3D surface that is rendered in perspective by a virtual camera. In such situation, the level of detail varies spatially depending on the distance of the 3D surface point from the camera. Here, proper anti-aliasing must compensate the foreshortening caused by a projective transformation. Next we present a simple example of anti-aliasing using the M-Net.

Let I be a checkerboard image, T be a *homography* mapping the pixel coordinates $x = (x_1, x_2)$ of the screen to the texture coordinates $u = (u_1, u_2)$ of I, and $f : \mathbb{R}^2 \times [0, N] \to \mathbb{R}$ be a M-net with N stages approximating a multiresolution of I.

Fig. 21(a) shows aliasing effects on the image I after applying it to the inverse of T. We avoid such a problem using the multiresolution given by f. The result is presented in Fig. 21(b).

The above procedure reduces aliasing at large distances. Specifically, we define the scale parameter $\lambda(x)$ for f at a pixel x using the Heckbert's formula [39]:

$$\lambda(x) = \max\left\{\sqrt{\left(\frac{\partial u_1}{\partial x_1}\right)^2 + \left(\frac{\partial u_2}{\partial x_1}\right)^2}, \sqrt{\left(\frac{\partial u_1}{\partial x_2}\right)^2 + \left(\frac{\partial u_2}{\partial x_2}\right)^2}\right\}.$$

Thus $\lambda(x)$ is the bigger length of the parallelogram generated by the vectors $\frac{\partial T}{\partial x_1}$ and $\frac{\partial T}{\partial x_2}$. We scale λ such that $\lambda\left([-1, 1]^2\right) \subset [0, N]$. Thus, $f(x, \lambda(x))$ is the desired level of detail $\lambda(x)$ of f.

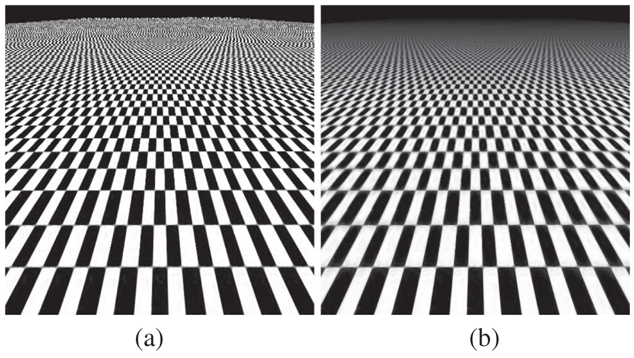
Fig. 21. Checkerboard in perspective: (a) point sampled texture rendition; (b) M-Net anti-aliased reconstruction.

Table 1
Comparison with SIREN and BACON.

Model	# Params (↓)	PSNR (↑)	# Levels
SIREN [24]	198K	**34.1** dB	1
BACON [26]	398K	33.1 dB	7
MR-Net (Ours)	**196K**	33.8 dB	7

5. Comparisons

In this section we compare the performance of MR-Net image representation with SIREN [24] and BACON [26].

First, we evaluate the quality of the original image fitting and the size of the representation for an instance of each model. Then, for MR-Net and BACON, the multiresolution models, we also evaluate the frequency spectra of the reconstructions in different scales. Finally, we proceed to a broader quantitative evaluation of the image reconstruction using the "Kodak Lossless True Color Image Suite" [30], for which we compute the average Peak Signal to Noise Ratio (PSNR), the model size and the average running time for some different configurations of each model.

5.1. Image fitting evaluation

We evaluate the capability of each model in the image fitting task using the "Cameraman" image, comparing the model size, given by the number of parameters of the model, and the reconstruction quality. We remark that to establish a fair comparison with SIREN we evaluate only the PSNR of the final full resolution image, which is 512 × 512. Table 1 summarizes the results.

The M-Net parameters are: 1 hidden layer per stage; 96 hidden features per layer in the first 6 stages, and 256 hidden features in the last stage; $\omega \in [4, 256]$ and trained with a Gaussian Pyramid of 7 levels. The model size has 195 648 parameters and the PSNR of the final image reconstruction is 33.8 dB. The model was trained for 300 epochs per stage.

For SIREN we employed the configuration of the image experiments and code released by the authors. The network parameters are: 3 hidden layers, 256 hidden features and ω_0 equal to 30. The model size has 198 400 parameters and it has only 1 level of detail. The PSNR of the reconstructed image is 34.1 dB after training for 2000 epochs. Fig. 23 shows a comparison between our M-Net and SIREN. Notice that, although M-Net presents a PSNR close to Siren's in this example, visually, it seems that it can better represent high frequency details.

For BACON we also based the configuration on their paper examples and code released by the authors, keeping 256 hidden features. However, we adjusted the number of hidden layers to 6 for it to output 7 levels of details as the M-Net in this setting.

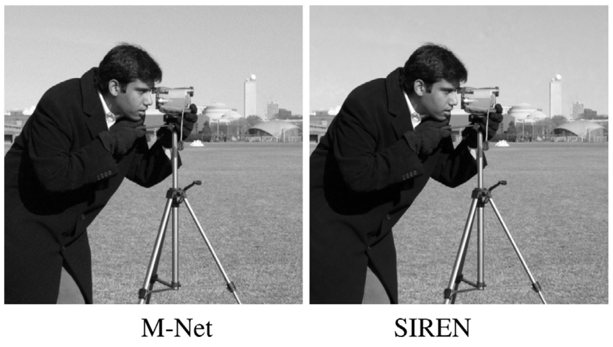

Fig. 22. Comparison of reconstructed images.

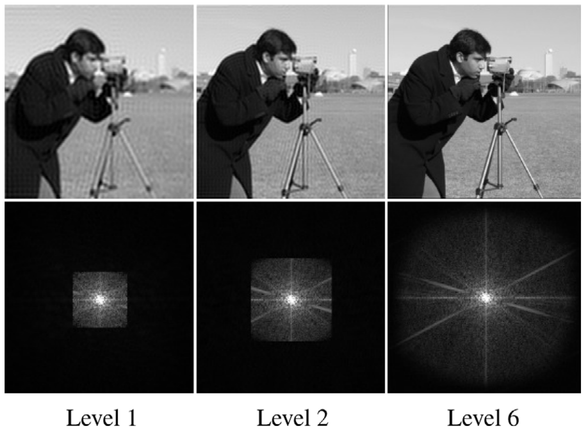

Fig. 23. BACON image reconstruction and frequency spectra.

Accordingly, the total number of parameters is 398 343. The PSNR of the original image level is 33.1 db.

Based on the experiments we conclude that MR-Net compared favorably in relation to BACON and SIREN, both in terms of representation size and quality of image reconstruction. The M-Net model is less than 50% of the size of BACON model, while our reconstruction of the final image presents a comparable (slightly higher) PSNR. Compared to SIREN, the M-Net is about the same size in number of parameters, but it encodes 7 levels of details in a single network, in contrast with 1 level for SIREN. Additionally, the PSNR of both models are comparable, and M-Net looks sharper.

5.2. Frequency spectra evaluation

Figs. 22 and 24 show the reconstructions and the frequency spectra for three levels of detail in the BACON and M-Net models respectively (1, 2, 6) and (3, 5, 7). Note that, as Bacon is not trained with multiresolution data, we select these levels to better match the frequency spectra between the representations.

BACON controls the frequency band for Level of Detail by truncating the spectrum. Fig. 22 shows that the center of the spectrum images of Levels 1, 2, 6 are all similar. This is analogous to applying a low-pass filter with non-ideal shape in the frequency domain, which results in an image with ringing effect (see the silhouettes propagating all over Fig. 22 (left)). M-Net does not present such artifacts. Fig. 24 resembles more faithfully what would be expected to be the process of sequentially applying Gaussian filters in a high-resolution image.

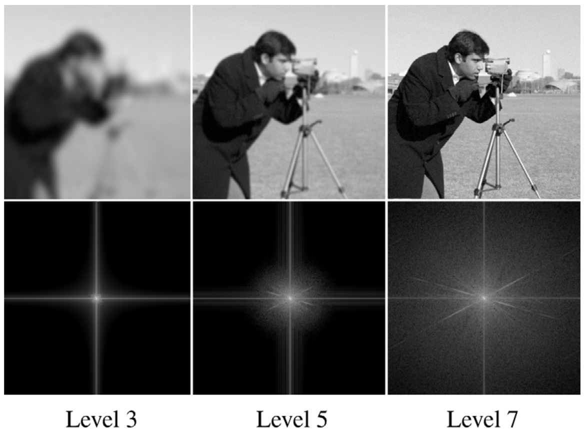

Fig. 24. M-Net image reconstruction and frequency spectra.

Table 2
Comparisons on Kodak Lossless True Color Image Suite. We present the training time relative to BACON$_{6hl\_256hf}$ which took 2433 s to train.

Model	# Params (↓)	PSNR (↑)	Time (↓)
SIREN$_{3hl\_256hf}$	198K	41.93	8.7%
SIREN$_{4hl\_256hf}$	264K	44.63	11.0%
SIREN$_{5hl\_256hf}$	330K	46.25	13.6%
SIREN$_{3hl\_512hf}$	790K	48.57	23.8%
BACON$_{5hl\_128hf}$	84K	27.42	36.2%
BACON$_{6hl\_128hf}$	101K	28.23	44.1%
BACON$_{4hl\_256hf}$	266K	33.69	60.8%
BACON$_{5hl\_256hf}$	332K	34.13	75.7%
BACON$_{6hl\_256hf}$	398K	34.31	100.0%
MNetCap$_{2stg\_256hf}$	117K	35.14	3.4%
MNet$_{7stg\_24\_256hf}$	193K	38.42	9.3%
MNet$_{7stg\_96\_256hf}$	196K	35.04	9.6%
MNet$_{6stg\_24\_288hf}$	197K	40.85	8.2%
MNetCap$_{128\_192\_256}$	195K	41.05	8.0%
MNetCap$_{3stg\_256hf}$	332K	45.46	11.4%

5.3. Kodak dataset evaluation

The "Kodak Lossless True Color Image Suite" is a set of 24 images, originally 768 × 512 in size, that were extensively used in the image processing literature. For each image, we center crop a square of 512 × 512 pixels, convert it to grayscale and evaluate the PSNR, model size and training time using different network configurations of each model. The average results over this dataset are summarized in Table 2.

We present the training time in a relative scale. All models were trained in a NVidia GeForce RTX 3080 with 10 GB of GPU memory. The longest average running time observed was 40 min and 33 s (2433 s) when training an instance of BACON with 6 hidden layers and 256 neurons per layer (Bacon_6hl_256). Therefore, we display this row as 100% and all others as a fraction of it.

We evaluate 4 SIREN instances: respectively 3, 4 and 5 hidden layers with 256 neurons per layer; and 3 hidden layers with 512 neurons per layer. For Bacon, we evaluate 5 instances: 5 and 6 hidden layers with 128 neurons per layer; and 4, 5, and 6 hidden layers with 256 neurons per layer.

All M-Net stages have depth 2, so we vary the number of stages and their width. As we are only evaluating the reconstruction at the final level, we trained M-Net with and without multiresolution data. Notice that we have more flexibility on the M-Net configuration as each MR-Module can have a different size. We decided to explore a few configurations with heterogeneous stages, choosing smaller modules for the coarsest levels, and bigger modules for the finest ones.

When training with a Gaussian pyramid, we evaluate $\text{MNet}_{7stg\_96\_256hf}$ with 7 stages and width in [96, 96, 96, 96, 96, 96, 256]; $\text{MNet}_{7stg\_24\_256hf}$ with 7 stages and width in [24, 32, 64, 96, 96, 160, 256]; and $\text{MNet}_{6stg\_24\_288hf}$ with 6 stages and width in [24, 32, 64, 96, 160, 288]. Although very different, these models have about the same number of total parameters. When training capacity based instances using only the original image, we evaluate $\text{MNetCap}_{2stg\_256hf}$ with 2 stages and width 256; $\text{MNetCap}_{3stg\_256hf}$ with 3 stages and width 256; and $\text{MNetCap}_{128\_192\_256}$ with 3 stages and width 128, 192, and 256 in each respective stage.

In general, inside each model category, as we increase the total number of parameters, the quality of the reconstruction also increases. Moreover, SIREN and M-Net models of the same size present a comparable performance in terms of PSNR, while BACON achieves lower values at the level of the original image. The exception occurs when we compare the M-Net models $\text{MNet}_{7stg\_24\_256hf}$ and $\text{MNet}_{7stg\_96\_256hf}$. Both models have 7 stages of multiresolution, but while the latter has a homogeneous architecture on all but the last level, the former distributes its capacity allocating less parameters in the initial levels. We think this heterogeneous distribution is better because on the coarsest levels we have lower frequencies and a less detailed signal to fit. When comparing $\text{MNet}_{7stg\_24\_256hf}$ and $\text{MNet}_{6stg\_24\_256hf}$, we see that as we remove one stage, and increase the width of the finest stages to keep a comparable model size, the performance of the model also increases.

Each M-Net model is trained for 2000 epochs per stage, while the others are trained for 5000 epochs. Although we train M-Net for more epochs, notice that due to our scheduled training scheme, we only update the parameters of a single stage each 2000 epochs. Besides that, when training with multiresolution data, the coarsest stages are trained with fewer samples. This way M-Net trains faster than the other models even when the number of parameters is bigger and the number of epochs too.

Training a 6 stages M-Net with 197 182 parameters ($\text{MNet}_{6stg\_24\_288hf}$) for 12 000 epochs takes a similar amount of time (199 s) as training a SIREN with 3 hidden layers and 198 401 parameters for 5000 epochs (211 s); and both are faster than training a BACON model with fewer parameters (BACON_{5hl_128hf} takes 881 s). The highest quality M-Net model in this evaluation, a M-Net capacity based with 331 523 parameters and 3 levels of details ($\text{MNetCap}_{3stg\_256hf}$), trained for 6000 epochs takes 278 s, which is only 15% of the time to train a BACON model of the same size (Bacon_{5hl_256hf} takes 1842 s) and is comparable to training a 20% smaller SIREN for 5000 epochs (Siren_{4hl_256hf} takes 267 s).

6. Considerations for other MR-Net variants

For the imaging application in this paper, we have only used the M-Net variant. In this section, we briefly present a comparison of M-Net with the others MR-Net variants.

For S-Net, we define a configuration composed of 3 stages, with 1000, 2000, and 4000 hidden features per layer respectively, which add to a total of 28 000 parameters. For the L-Net, the configuration follows 3 stages with 2 hidden layers with 40, 60, and 80 neurons for each layer. This L-Net version has an overall number of parameters of 24.280. For the last variant, the M-Net, we set 3 stages with 1 hidden layer each. Each stage has 30, 40, and 80 hidden features per layer respectively where the final number of parameters was 23.100. During training the S-Net, L-Net, and M-Net were trained at the maximum of 4000, 2000, and 2000 epochs respectively and we test 10 different images with resolution 128 × 128 for training. As we can see in Table 3, on average, considering the PSNR, the M-Net outperforms the other variants when reconstructing images.

Nonetheless, here it is appropriate to make a few considerations about the S-Net and L-Net variants.

Table 3
Comparison between the MR-Net variants.

Model	# Params (↓)	PSNR (↑)
S-Net	28 000	42.1 dB
L-Net	24 280	47.8 dB
M-Net	**23 100**	**51.1** dB

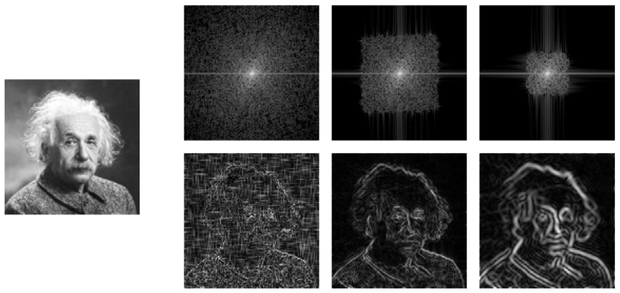
Fig. 25. S-Net gradient magnitude and frequency spectra.

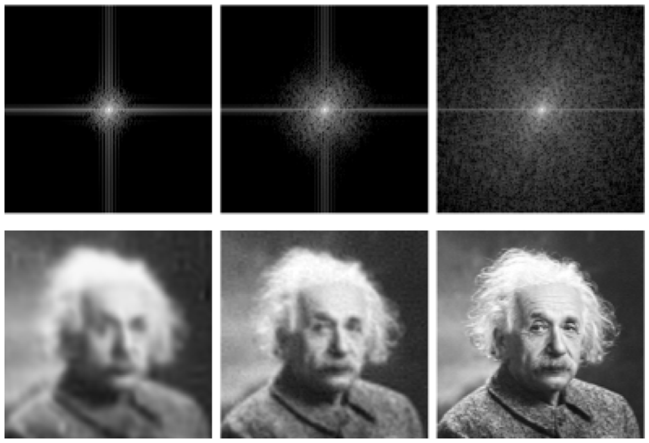
Fig. 26. L-Net reconstruction and frequency spectra.

The S-Net represents the image as a weighted sum of sine functions. In that sense, S-Net is equivalent to BACON and other Multiplicative Filter Networks based on sinusoidal atoms. As such, it is amenable to represent periodic visual patterns. Fig. 25 presents the gradient magnitude and frequency spectra from an image predicted by S-net, where it has three stages.

The L-Net image representation comprises a sum of level-of-detail stages given by independent MR-Modules. The relation of this representation with the Laplacian Pyramid makes it suitable for image operations in the gradient domain. Fig. 26 shows the L-Net reconstruction for the same image as S-Net and the frequency spectra for both L-Net stages.

7. Closing remarks

In this last section we close the paper with an assessment of our results, as well as, its limitations, and a discussion of future directions for our research.

7.1. Limitations

We found that choosing appropriate values for the ω parameter, which defines the spatial frequency of the network first layer, is important to achieve proper results. When using a shallow sinusoidal network such as the S-Net, we can use the Nyquist frequency as a direct reference to pick the frequency intervals at each stage. However, a deep sinusoidal network such as the M-Net can learn higher frequencies that were not present in the initialization.

In our experiments, we have determined the ω values for initialization of the frequency bands empirically, testing values below the Nyquist frequency as described in Section 2.6.1. To better harness the power of sinusoidal neural networks, it is important to develop mathematical theories to understand how the composition of sine functions introduces new frequencies based on the initialization of the network. In future works we intend to investigate the use of the results in [34] to compute or bound these frequencies.

7.2. Ongoing and future work

We have described a flexible method of working with signals in multiresolution in terms of multiple ways of preparing the input data, defining the MR-Net subclasses, and training multi-stage networks. In the applications presented in Section 4, we have explored a subset of this framework, showing cases where it improves upon existing state of the art techniques. Regarding other aspects of this groundwork, we have ongoing research on signal reconstruction from stochastic sampling, and training of L-Net models using the Laplacian pyramid, which may lead to novel imaging applications. Some of the motivation and experiments with 1D signals in these directions are documented in Velho et al. [40].

In terms of future work, we plan to expand this research in two main directions. On one hand, we would like to explore the MR-Net architecture for other image applications including super-resolution, operations in the gradient domain, generation of periodic and quasi-periodic patterns, as well as image compression. On the other hand, we would like to extend the MR-Net representation to other media signals in higher dimensions, such as video, volumes, and implicit surfaces.

CRediT authorship contribution statement

Hallison Paz: Formal analysis, Software, Validation, Conceptualization, Investigation, Methodology, Writing – review & editing. **Daniel Perazzo:** Software, Validation, Conceptualization, Investigation, Methodology, Writing – review & editing. **Tiago Novello:** Formal analysis, Validation, Conceptualization, Investigation, Methodology, Writing – review & editing. **Guilherme Schardong:** Validation, Conceptualization, Investigation, Methodology, Writing – review & editing. **Luiz Schirmer:** Validation, Conceptualization, Investigation, Methodology, Writing – review & editing. **Vinícius da Silva:** Validation, Conceptualization, Investigation, Methodology, Writing – review & editing. **Daniel Yukimura:** Software, Conceptualization, Investigation, Methodology, Writing – review & editing. **Fabio Chagas:** Validation, Conceptualization, Investigation, Methodology, Writing – review & editing. **Hélio Lopes:** Supervision, Conceptualization, Investigation, Methodology, Writing – review & editing. **Luiz Velho:** Supervision, Conceptualization, Investigation, Methodology, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

The authors are supported by grants from CNPq, CAPES and FAPERJ.

References

[1] LeCun Yann, Bengio Yoshua, Hinton Geoffrey. Deep learning. Nature 2015;521(7553):436–44.
[2] Goodfellow Ian, Bengio Yoshua, Courville Aaron. Deep learning. MIT Press; 2016.
[3] LeCun Yann, Bengio Yoshua. Convolutional networks for images, speech, and time series. In: The handbook of brain theory and neural networks. MIT Press; 1998, p. 255–8.
[4] Li Zewen, Liu Fan, Yang Wenjie, Peng Shouheng, Zhou Jun. A survey of convolutional neural networks: analysis, applications, and prospects. IEEE Trans Neural Netw Learn Syst 2021.
[5] Shamsaldin Ahmed S, Fattah Polla, Rashid Tarik A, Al-Salihi Nawzad K. A study of the convolutional neural networks applications. UKH J Sci Eng 2019;3(2):31–40.
[6] Xie Yiheng, Takikawa Towaki, Saito Shunsuke, Litany Or, Yan Shiqin, Khan Numair, Tombari Federico, Tompkin James, Sitzmann Vincent, Sridhar Srinath. Neural fields in visual computing and beyond. In: Computer graphics forum, Vol. 41. Wiley Online Library; 2022, p. 641–76.
[7] Dupont Emilien, Goliński Adam, Alizadeh Milad, Teh Yee Whye, Doucet Arnaud. Coin: Compression with implicit neural representations. 2021, arXiv preprint arXiv:2103.03123.
[8] Czerkawski Mikolaj, Cardona Javier, Atkinson Robert, Michie Craig, Andonovic Ivan, Clemente Carmine, Tachtatzis Christos. Neural knitworks: Patched neural implicit representation networks. 2021, arXiv preprint arXiv:2109.14406.
[9] Chen Yinbo, Liu Sifei, Wang Xiaolong. Learning continuous image representation with local implicit image function. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021, p. 8628–38.
[10] Park Jeong Joon, Florence Peter, Straub Julian, Newcombe Richard, Lovegrove Steven. Deepsdf: Learning continuous signed distance functions for shape representation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019, p. 165–74.
[11] Mildenhall Ben, Srinivasan Pratul P, Tancik Matthew, Barron Jonathan T, Ramamoorthi Ravi, Ng Ren. Nerf: Representing scenes as neural radiance fields for view synthesis. Commun ACM 2021;65(1):99–106.
[12] Marr David. Vision: A computational investigation into the human representation and processing of visual information. New York, NY, USA: Henry Holt and Co., Inc.; 1982.
[13] Burt Peter J, Adelson Edward H. The Laplacian pyramid as a compact image code. In: Readings in computer vision. Elsevier; 1987, p. 671–9.
[14] Williams Lance. Pyramidal parametrics. SIGGRAPH Comput Graph 1983;17(3):1–11. http://dx.doi.org/10.1145/964967.801126.
[15] Thies J, Zollhöfer M, ner M Nieß. Deferred neural rendering: Image synthesis using neural textures. ACM Trans Graph 2019;38(4). http://dx.doi.org/10.1145/3306346.3323035.
[16] Bracewell Ronald Newbold, Bracewell Ronald N. The fourier transform and its applications, Vol. 31999. McGraw-Hill New York; 1986.
[17] Bhaskaran Vasudev, Konstantinides Konstantinos. Image and video compression standards: algorithms and architectures. Springer Science & Business Media; 1997.
[18] Marcellin Michael W, Gormish Michael J, Bilgin Ali, Boliek Martin P. An overview of JPEG-2000. In: Proceedings DCC 2000. Data compression conference. IEEE; 2000, p. 523–41.
[19] Ahmed N, Natarajan T, Rao KR. Discrete cosine transform. IEEE Trans Comput 1974;C-23(1):90–3. http://dx.doi.org/10.1109/T-C.1974.223784.
[20] Mallat Stephane G. A theory for multiresolution signal decomposition: the wavelet representation. IEEE Trans Pattern Anal Mach Intell 1989;11(7):674–93.
[21] Antonini Marc, Barlaud Michel, Mathieu Pierre, Daubechies Ingrid. Image coding using wavelet transform. IEEE Trans Image Process 1992;1(2):205–20.
[22] Froment Jacques, Mallat Steéphane. Second generation image coding and wavelet transform. In: Proceedings of the first world congress on world congress of nonlinear analysts '92, Vol. II. WCNA '92, USA: Walter de Gruyter & Co.; 1996, p. 1923–32.
[23] Parascandolo G, Huttunen H, Virtanen T. Taming the waves: sine as activation function in deep neural networks. 2017.
[24] Sitzmann Vincent, Martel Julien NP, Bergman Alexander W, Lindell David B, Wetzstein Gordon. Implicit neural representations with periodic activation functions. In: Proc. NeurIPS. 2020.
[25] Fathony Rizal, Sahu Anit Kumar, Willmott Devin, Kolter J Zico. Multiplicative filter networks. In: International conference on learning representations. 2020.

[26] Lindell D, Veen D Van, Park J, Wetzstein G. BACON: Band-limited coordinate networks for multiscale scene representation. In: Proceedings of CVPR. 2022.

[27] Müller Thomas, Evans Alex, Schied Christoph, Keller Alexander. Instant neural graphics primitives with a multiresolution hash encoding. ACM Trans Graph 2022;41(4):102:1–102:15. http://dx.doi.org/10.1145/3528223.3530127.

[28] Martel Julien NP, Lindell David B, Lin Connor Z, Chan Eric R, Monteiro Marco, Wetzstein Gordon. ACORN: Adaptive coordinate networks for neural scene representation. ACM Trans Graph (SIGGRAPH) 2021;40(4).

[29] Paz Hallison, Novello Tiago, Silva Vinicius, Schardong Guilherme, Schirmer Luiz, Chagas Fabio, Lopes Helio, Velho Luiz. Multiresolution neural networks for imaging. In: 2022 35th SIBGRAPI conference on graphics, patterns and images (SIBGRAPI), Vol. 1. 2022, p. 174–9. http://dx.doi.org/10.1109/SIBGRAPI55357.2022.9991765.

[30] Kodak lossless true color image suite. 2023, http://r0k.us/graphics/kodak/, Accessed: 2023-01-30.

[31] Rosenfeld A. Multiresolution image processing and analysis. Springer series in information sciences, Springer Berlin Heidelberg; 2013, URL https://books.google.com.br/books?id=ZGirCAAAQBAJ.

[32] Lindeberg Tony. Scale-space theory: A basic tool for analyzing structures at different scales. J Appl Stat 1994;21(1–2):225–70.

[33] Velho Luiz, Frery Alejandro C, Gomes Jonas. Image processing for computer graphics and vision. Springer Science & Business Media; 2009.

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

[35] Rahaman Nasim, Baratin Aristide, Arpit Devansh, Dräxler Felix, Lin Min, Hamprecht Fred A, Bengio Yoshua, Courville Aaron C. On the spectral bias of neural networks. In: International conference on machine learning. 2018.

[36] Cook Robert. Stochastic sampling in computer graphics. ACM Trans Graph 1986;5:51–72. http://dx.doi.org/10.1145/7529.8927.

[37] Bridson Robert. Fast Poisson disk sampling in arbitrary dimensions. ACM SIGGRAPH 2007. http://dx.doi.org/10.1145/1278780.1278807.

[38] Smith Alvy Ray. A biography of the pixel. Boston: MIT Press; 2021.

[39] Heckbert Paul S, et al. Texture mapping polygons in perspective. Technical report, NYIT Computer Graphics Lab; 1983.

[40] Velho Luiz, Paz Hallison, Novello Tiago, Yukimura Daniel. Multiresolution neural networks for multiscale signal representation. Technical report, VISGRAF Lab; 2022.