Multiresolution Neural Networks for Multiscale Signal Representation

Luiz Velho, Hallison Paz, Tiago Novello, Daniel Yukimura

Abstract

In this document we introduce a new scheme for signal representation in multi-scale using multi-resolution neural networks.

1 Representing Signals

Some phenomena that occur in our everyday life, such as sound, images, etc, are characterized by functions. To each point in space or each instant in time the function produces a value. These functions are usually called *signals*.

A *representation* of a normed space of functions $\mathcal{F}$ is an operator $R\colon \mathcal{F} \to \mathcal{S}$ into the normed space of sequences $(c_j)_{j\in\mathbb{Z}}$ of numbers. For a given function $f \in \mathcal{F}$, its representation $R(f)$ is a sequence

$$R(f) = (f_j)_{j\in\mathbb{Z}} \ \in \mathcal{S}.$$

R is called the *representation operator*. In general it is natural to demand that R preserves norms. That is

$$\|R(f)\| = \|f\|,$$

or that R satisfies some stability condition, such as

$$\|R(f)\| \leq C\,\|f\|.$$

When R is linear and continuous, we have a linear representation.

The most important examples of representations occurs when the space of functions $\mathcal{F}$ is a subspace of the space $\mathbf{L}^2(\mathbb{R})$, of square integrable functions (finite energy),

$$\mathbf{L}^2(\mathbb{R}) = \{f\colon \mathbb{R} \to \mathbb{R} \ ; \ \int_{\mathbb{R}} |f(t)|^2 dt < \infty\},$$

and the representation space $\mathcal{S}$ is the space ℓ^2 of the square summable sequences,

$$\ell^2 = \left\{ (x_i)_{i\in\mathbb{Z}} \ ; \ x_i \in \mathbb{C}, \text{and} \sum_{i=-\infty}^{+\infty} |x_i|^2 < \infty \right\}.$$

When the representation operator is invertible, we can reconstruct f from its representation sequence: $f = R^{-1}\big((f_i)_{i\in\mathbb{Z}}\big)$. In this case, we have an *exact representation*, otherwise it is an *approximate representation.*

Figure 1 shows the general function representations scheme. The sequence of functions e_j is a "basis" of $\mathcal{F}$ as described in the next section.

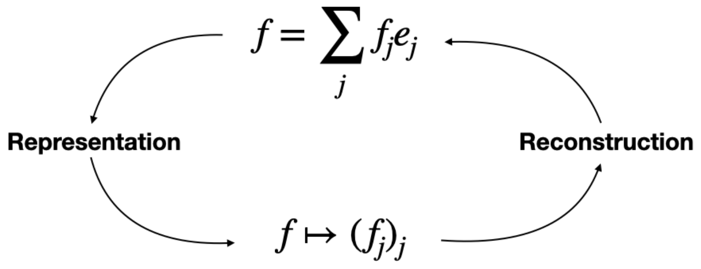

Figure 1: General Function Representation Scheme

1.1 Function Representation Schemes

How we define the representation operator gives different representation schemes. The most relevant ones are representation by Basis, Frames, and Dictionaries.

1.1.1 Basis

A natural technique to obtain a representation of a space of functions consists in constructing a basis of the space. A set $B = \{e_j;\, j \in \mathbb{Z}\}$ is a basis of the function space $\mathcal{F}$ if the vectors e_j are linearly independent, and for each $f \in \mathcal{F}$ there exists a sequence $(\alpha_j)_{j\in\mathbb{Z}}$ of numbers such that

$$f = \sum_{j\in\mathbb{Z}} \alpha_j e_j. \tag{1}$$

The above equality means the convergence of partial sums of the series in the norm of the space $\mathcal{F}$:

$$\lim_{n\to\infty} \left\| f - \sum_{j=-n}^{n} \alpha_j e_j \right\| = 0.$$

We define the representation operator by

$$R(f) = (\alpha_j)_{j\in\mathbb{Z}}.$$

Equation (1) reconstructs the function f from the representation sequence $(\alpha_j)_{j\in\mathbb{Z}}$.

1.1.2 Frames

The basic idea of representing functions on a basis, consists in decomposing it using a countable set of simpler functions.

The existence of a complete orthonormal set, and its construction is in general a very difficult task. Therefore, it is important to obtain collections of functions $\{\varphi_n; n \in \mathbb{Z}\}$ which do not constitute necessarily an orthonormal set or are not linearly independent, but can be used to define a representation operator. One such collection is constituted by the *frames*.

One important example is the *Riesz basis*. If a collection $\mathcal{B} = \{\varphi_n; n \in \mathbb{Z}\}$ is a frame, and the functions φ_j are linearly independent, we have a *Riesz basis*. Therefore if $\{e_n\}$ is a Riesz basis, for any $f \in \mathcal{H}$ we have

$$A\|f\|^2 \leq \sum_n |\langle f, e_n\rangle|^2 \leq B\|f\|^2 .$$

For each function $f \in \mathbf{L}^2(\mathbb{R})$, we define its representation

$$\sum_k \langle f, e_k\rangle e_k,$$

on the Riesz basis $\{e_k\}$.

Note that, differently from the basis representation, the representation by frames is not unique.

1.1.3 Dictionaries

A dictionary in a function space H is a family $\mathcal{D} = (g_\lambda)_{\lambda\in\Gamma}$ of vectors in H such that $\|g_\lambda\| = 1$. A representation of a function in a dictionary $\mathcal{D}$ is a decomposition

$$f = \sum_{\lambda\in\Gamma} \alpha_\gamma g_\gamma,$$

such that $(\alpha_\gamma) \in \ell^2$.

The rationale behind the method consists in constructing extensive dictionaries and devising optimal representation techniques that allow us to represent a function using a minimum of words from the dictionary. Therefore, representation using dictionaries allows us the use of a great heterogeneity in the reconstruction functions, which makes the representation/reconstruction process very flexible.

Note that distinct functions use different dictionary vectors in their representation which makes the representation process non-linear.

One basic strategy to compute a representation for a function f is described below.

Let f_M be the projection of f over the space generated by M vectors from the dictionary, with index set I_M:

$$f_M = \sum_{m\in I_M} \langle f, g_m\rangle g_m.$$

The error e_M in the approximation is the sum of the remaining coefficients

$$e_M = f - f_M{}^2 = \sum_{m \notin I_M} |\langle f, g_m \rangle|^2.$$

To minimize this error the indices in I_M must correspond to the M vectors that have the largest inner product amplitude $|\langle f, g_m \rangle|$. These are the vectors that have a better correlation with f, that is, the vectors that best match the features of f. Certainly the error is smaller than the error resulting from a linear approximation where the decomposition vectors of the representation do not vary with f.

The above ideas lead to an algorithm to compute the representation.

1.2 Representation Domains

As discussed in the previous section, signals take values on spatial or temporal domains. Thus these functions are naturally defined on these spaces. However, in order to devise good representation techniques we must develop tools that enable us to locate distinguished features of a function. The most traditional of these tools is the Fourier Transform.

1.2.1 Spatial and Frequency Domains

The Fourier transform is an invertible linear operator on $\mathbf{L}^2(\mathbb{R})$. Nevertheless, in the applications we have a very interesting interpretation of it. A function $f : \mathbb{R} \to \mathbb{R}$, can be interpreted as associating to each value of t in the spatial domain $\mathbb{R}$ some physical magnitude $f(t)$. When we compute the Fourier transform of f, we obtain another function $\hat{f}(s)$ defined on $\mathbf{L}^2(\mathbb{R})$. In this case, for each value of the parameter $s \in \mathbb{R}$ the value $\hat{f}(s)$ represents the frequency density s in f. We interpret this by saying that $\hat{f}$ is a representation of the function f in the frequency domain. In summary, the Fourier transform changes a function from the spatial to the frequency domain.

Often, when the spatial domain has dimension 1 it is common to interpret the variable t as time and call the spatial domain by the name of *time domain*.

Note that describing a function on the frequency domain allow us to obtain the frequency contents of the function. The frequency contents is closely related with the features carried by the function. These features are important elements to obtain good function representations.

Fourier Series and Spectral Representation

Consider a periodic function with period $L > 0$, that is, $f(t + L) = f(t)$. We denote by $L^2_T(\mathbb{R})$ the space of periodic functions of period T which are square integrable. The theory of Fourier series says that f can be decomposed as

$$f(s) = \sum_{j=-\infty}^{+\infty} a_j e^{i2\pi\omega_j s}, \quad a_j \in \mathbb{R}, \tag{2}$$

where $\omega_j = j/T$ is a constant. This decomposition of a periodic function f is called the *Fourier series* of f. It is well known that the family $\{e^{i2\pi\omega_j s},\ j \in \mathbb{Z}\}$ is a complete orthonormal set of the space $\mathbf{L}^2_T(\mathbb{R})$. Therefore equation (2) is an orthogonal basis representation of the function f.

In conclusion, the Fourier series shows that any periodic function can be decomposed as an infinite sum of periodic functions (sines and cosines). This decomposition makes it easy an analysis of the frequencies present on the function f: There exists a fundamental frequency ω, and all of the other frequencies are integer multiples ωj, $j \in \mathbb{Z}$, of this fundamental frequency.

The coefficient a_j in the equation (2) of the Fourier series measures the amplitude of the frequency component ω_j on the function f. In particular, if $a_j = 0$, this frequency is not present in the function. This frequency amplitude a_j is computed using the equation

$$a_j = \int_0^L f(u) e^{i2\pi\omega_j u} du, \tag{3}$$

where L is the period of the function. Note that equation (2) completely characterizes the function f by its frequencies. In other words, we have an exact representation of the periodic function f.

When the function f is not periodic the Fourier transform, gives us an analysis of the frequencies of the function f. Note that the Fourier transform is not discrete as in the case of the Fourier series, for periodic functions. Therefore we do not have a tool that allow us to represent and reconstruct arbitrary functions as in the case of the Fourier series.

1.2.2 Scale and Detail Domains

We perceive the world through a multiscale mechanism. First we use a coarse scale to recognize the object, then we use finer scales in order to discover its distinct properties in detail.

Therefore, it is natural that we look for multiscale representation of functions. That is, we are interested in obtaining a family of representations that could represent the function at distinct scales. At the same time we need techniques that allow us to change between representations on different scales.

This can be achieved by using nested representation spaces. Consider a sequence of closed subspaces $\{V_j\}_{j\in\mathbb{Z}}$ such that

$$\cdots V_{j+1} \subset V_j \subset V_{j-1} \cdots, \quad \forall \in \mathbb{Z}.$$

And a family of representation operators $P_j \colon V \to V_j$ such that

$$\|v - P_j(v)\| \leq C \inf_{u\in V_j} \|v - u\|, \tag{4}$$

where C independs on j. The proximity condition of equation (4) is trivially satisfied if V_j is a closed subspace of a Hilbert space, and P_j is the orthogonal projection onto V_j.

Note that from equation (4) we have

$$\|P_j(v)\| \le c\,\|v\| \tag{5}$$

$$P_j(v) = v, \quad \forall v \in V_j \text{ (i.e. } P_j^2 = P_j) \tag{6}$$

Intuitively each representation space V_{j-1} contains more details (finer scale) than the space V_j. This can be stated precisely by introducing the operator $Q_j\colon V \to V_{j-1}$, defined by

$$Q_j(v) = P_{j+1}(v) - P_j(v).$$

If $W_j = \text{Image}(Q_j) = Q_j(V_j)$, it follows that

$$V_{j-1} = V_j + W_j. \tag{7}$$

That is, $Q_j(v)$ is the detail we must add to the representation space V_j to obtain V_{j+1}. For this reason, Q_j is called a *refinement operator*.

Iterating equation (7) we have the decomposition equation

$$V_J = V_{j_0} + (W_{j_0} + \cdots + W_{J+1}), \tag{8}$$

which says that a finer representation can be obtained from a finer one, by adding details. Equation (8) can be rewritten using operators:

$$P_j(v) = P_{j_0}(v) + \sum_{j=J+1}^{j_0} Q_j(v). \tag{9}$$

In order to be able to decompose any element v of V we impose some additional hypothesis on the representation operators P_j:

1. $P_j(v) \to v \quad \text{if} \quad j \to -\infty$;
2. $P_j(v) \to 0 \quad \text{if} \quad j \to +\infty$.

From the proximity condition of equation (4), these two conditions are equivalent to

1. $\bigcup_{j\in\mathbb{Z}} V_j$ is dense on V;
2. $\bigcap_{j\in\mathbb{Z}} = \{0\}$.

By taking the limit, $j_0 \to +\infty$ and $j \to -\infty$, in (9) we have

$$v = \sum_{j\in\mathbb{Z}} Q_j(v), \quad \forall v \in V.$$

That is, any vector v is the sum of all of its details in the different representations.

2 Sampling and Reconstruction

In this section we will consider the problem of function representation in the context of sampling and reconstruction schemes.

2.1 Sampling Techniques

Sampling techniques can be interpreted as means of obtaining the representation of a function from its values. There are two main sampling techniques: classical (or uniform) and stochastic sampling.

2.1.1 Classical Sampling

In the *classical sampling* scheme we have a uniform sampling partition of the real numbers with interval length Δt. This length is called the *sampling period.* The discretized representation is given by

$$f_d(t) = \sum_{k \in \mathbb{Z}} f(k\Delta t)\delta(t - k\Delta t).$$

The above scheme poses the question: *Is it possible to obtain an exact representation of a function using point sampling?*

We will see that the answer is positive if we restrict the function f and at the same time impose conditions on the sampling rate. Initially we will demand the following conditions:

- The point sampling process is uniform. That is, the sampling intervals $[t_k, t_{k+1}]$ have the same length $\Delta t = t_{k+1} - t_k$.
- The Fourier transform $\hat{f}$ of the function f, assumes zero values outside a bounded interval $[-\Omega, \Omega]$ of the frequency domain. We say that $\hat{f}$ has *compact support.*

Using the above restrictions, we have the classical:

Theorem 1 (Shannon-Whittaker)**.** *Consider a function $f\colon \mathbb{R} \to \mathbb{R}$ with compact support,* $\operatorname{supp}(\hat{f}) \subset [-\Omega, \Omega]$, *and a uniform partition t_i, $i \in \mathbb{Z}$ of the real numbers $\mathbb{R}$ such that $2\Delta t \leq \Omega$. Then f can be reconstructed (exactly) from its point sampling representation $(f(t_i))_{i\in\mathbb{Z}}$. The reconstruction equation is given by*

$$f(t) = \sum_{k=-\infty}^{+\infty} f(\frac{k}{2\Omega})\frac{\sin \pi(2\Omega t - k)}{\pi(2\Omega t - k)}. \tag{10}$$

The inequality $2\Delta x \leq \Omega$ is called *Nyquist limit.* It says that we must take at least 2 samples for each complete cycle of maximum frequency occurring in the function.

The result of the theorem is very intuitive. The hypothesis of the theorem says:

1. The function f should not have very high frequencies, $\text{supp}\,(\hat{f}) \subset [-\Omega, \Omega]$;
2. We take samples of f sufficiently close, such that we have at least 2 samples for a complete cycle of maximum frequency $(2\Delta t \leq \Omega)$;

The conclusion of the theorem is: f can be exactly reconstructed interpolating its samples using equation (10).

A function f such that $\hat{f}$ has compact support is called a *band limited function*, because it posses frequencies within a limited interval (band) of the frequency domain.

In this context, it can be shown that if we take e as the sinc function

$$e_\Omega(t) = \frac{\sin \pi 2\Omega t}{\pi 2\Omega t},$$

the set $\{e_\Omega(t - k\Delta t)\}$ is an orthogonal basis of the space

$$L^2_\Omega(\mathbb{R}) = \{f \in L^2(\mathbb{R})\ ;\ \text{supp}(\hat{f}) \subset [-\Omega, \Omega]\}.$$

Moreover, the reconstruction equation (10) is the projection of f on this basis. In sum, the problem of point sampling and reconstruction is reduced to the problem of representation on an orthogonal basis.

If a signal f is not band-limited, that is, $f \notin L^2_\Omega(\mathbb{R})$, and it is represented using point sampling with period $\Delta t \leq \Omega/2$, and we reconstruct the sampled signal using the reconstruction equation (10) of Shannon-Whittaker, we obtain a function $\tilde{f} \in L^2(\mathbb{R})$ such that $\left\| \tilde{f} - f \right\|$ is minimized. In fact, $\tilde{f}$ is the orthogonal projection of f on $L^2_\Omega(\mathbb{R})$.

2.1.2 Stochastic Sampling

Sometimes modelling signal as functions is unsatisfactory and the use of stochastic processes is more realistic. Here we will introduce shift-invariant stochastic processes. The model is inspired by band-limited stochastic processes and shift-invariant spaces. A stochastic process is said to be shift-invariant, if its auto-correlation function belongs to a shift-invariant signal space. It is well known that a band-limited signal space is equivalent to a shift-invariant signal space with generator $\psi = \text{sinc}$. Hence the band-limited stochastic processes are a special case of the shift-invariant stochastic processes.

Signals can be reconstructed from its ideal and average samples taken on a relatively separated set with small gap for shift-invariant stochastic processes as stated below.

Consider a real *stochastic process* $\{X(t, \omega) : t \in \mathbb{R}\}$ defined on a probability space(A, Ω, P). The basic assumptions on the process are:

(a) $E|X(t, \cdot)|^2 < \infty, \forall_t \in \mathbb{R}$

(b) $\{X(t, \omega)\}$ is stationary, i.e. $R_X(t + u, t) = R_X(u)$ is independent of t for all $t \in \mathbb{R}$, where R_X stands for the *autocorrelation* function given by

$$R_X(t, t') = E[X(t, \omega)X(t', \omega)] \tag{11}$$

A wide sense stationary process is said to be *shif-invariant* if its autocorrelation function $R_X(u)$ belongs to a shift-invariant signal space, that is

$$R_X(t,t') = \sum_{k\in\mathbb{Z}} c(k)\psi(t-t'-k). \tag{12}$$

Taking $\psi = \text{sinc}$, then the shift-invariant stochastic process is equivalent to a band-limited stochastic process.

Furthermore, a shift-invariant space is a space of functions on $\mathbb{R}$ of the form

$$V^2(\psi) = \Big\{\sum_{k\in\mathbb{Z}} c(k)\psi(\cdot,-k) : c \in l^2\Big\} \tag{13}$$

where ψ is the generator.

For continuous and stable generator ψ, there exists a unique function $\tilde{\psi}$ such that $\{\widetilde{\psi}(\cdot,-k) : k \in \mathbb{Z}\}$ is a Riesz basis for $V^2(\psi)$ with ψ and $\tilde{\psi}$ satisfying the biorthogonality. So $K(x,y)$ is a reproducing kernel of $V^2(\psi)$. A reconstruction formula is presented by

$$f(x) = \sum_{j\in\mathbb{Z}} \langle f, K(x_j,\cdot)\rangle \widetilde{K(x_k,\cdot)} = \sum_{j\in\mathbb{Z}} f(x_j)\widetilde{K(x_k,\cdot)} \tag{14}$$

Then, $f \in V^2(\psi)$ can be recovered from its weighted average samples $\{\langle f, \phi_{x_j}\rangle : x_j \in X\}$.

Theorem 2 (Xian-Li)**.** *Let $X(t,\omega)$ be a shift-invariant stochastic process, the generator ψ be continuous and stable for $V^2(\psi)$, ψ satisfying $\psi(x) = O(1+|x|)^{-\beta}$ for some $\beta > 1$ and $X := \{x_j : j \in \mathbb{Z}\}$ be a sampling set for $V^2(\psi)$. Then there exist $\{K(x_k,\cdot)\}$ such that*

$$X(t,\omega) = \sum_{k\in\mathbb{Z}} X(x_k,\omega) > K(\widetilde{x_k,t}),$$

where the convergence is in mean-square.

Theorem 2, applies in the case of ideal sampling for shift-invariant stochastic processes. It can be extended to reconstruction from weighted average sampling [1].

2.2 Reconstruction Methods

Reconstruction methods recover the function from its representation given by the sampling techniques as presented in the previous subsection.

2.2.1 Reconstruction Using Basis

When we have an orthonormal basis $\{\phi_j\}$ of the space $\mathbf{L}^2(\mathbb{R})$, the representation of f in this basis is given by

$$f = \sum_n \langle f, \phi_n \rangle \phi_n.$$

In this case we have the representation sequence

$$f \mapsto (\langle f, \phi_n \rangle)_{n \in \mathbb{Z}}.$$

Note that if $\phi_n(x) = \delta(x - n)$, then

$$\langle f, \phi_n \rangle = \langle f, \delta(x - n) \rangle = f(n).$$

That is, the elements of the representation sequence are samples of the function f. Recall that this is the case of classical sampling presented in Section 2.1.1.

Other cases, for frames and dictionaries have a similar interpretation.

2.2.2 Kernel Regression

Kernel regression is a reconstruction method that, given a set of values $y_i = f(x_i)$, for $i = 1, \ldots n$, constructs an estimate $\hat{f}$ of the underlying function f as:

$$\hat{f}_w(x) = \sum_{i=1}^{n} w_i k(x - x_i), \tag{15}$$

where k is the kernel function and w_i are the weights corresponding to the kernel centered at the sampling points x_i.

The kernel is a function with the following properties:

1. k is symmetrical, i.e. $k(u) = k(-u)$
2. k has norm 1, i.e., $\int_{-\infty}^{\infty} k(u)du = 1$
3. k is positive, i.e., $k(u) \geq 0$, for $-\infty < u < \infty$

A typical example of kernel function is the Gaussian

$$k(x) = \frac{1}{\sqrt{2\pi}} \exp(-x^2/2)$$

The reconstruction method gives the optimal weights in a least-squares sense:

$$\min_w \sum_i \left\| f(x_i) - \hat{f}_w(x_i) \right\|^2$$

Intuitively, $k(x - x_i)$ is a smoothing weight function that represents the "similarity" between the two points x and x_i.

We remark that there is a connection between kernel regression and stochastic sampling discussed in subsection 2.1.2.

3 Neural Implicit Signal Representation

In this section we study a representation for signals using neural networks. We are particularly interested in signals as *media objects*. In that sense, they are given by functions in dimensions one, two and three.

Example of one-dimensional signal is a sound. Examples of two-dimensional signals are images and surfaces. Examples of three-dimensional signals are videos and 3D scenes. Higher dimensional signals exist but are not going to be considered here.

Media objects can be defined either in the *time domain* (sound), or in *spatial domain* (images, 3D scenes), and even in *space-time domain* (video, animations).

3.1 Signal Representation Modalities

In order to represent media objects of dimension greater than one it is possible to adopt two modalities: *parametric* or *implicit*.

In the *parametric form*, points belonging to the object are given directly by a mapping function or parameterization. These mappings relate a space of parameters to the object such that there is a correspondence between points in these two spaces.

In the *implicit form*, points belonging to the object are given indirectly through a point-membership classification function. This function defines the relation of points in the ambient space with the object.

These two forms are in a sense complementary. It is important to notice that the parametric form is a *intrinsic* description while the implicit form is an *extrinsic* one.

Intuitive understanding of the differences between the parametric and implicit forms is gained by means of the simple example of a curve in the plane:

The parametric form describes the curve by the equation $f(t) = p$, where p is a point with coordinates (x, y):

$$f(t) = (x, y), \quad t \in [0, 1].$$

This parameterization maps the interval $[0, 1]$ of the real line onto points of the curve. This allows us to directly enumerate all the points p_t by varying the parameter t.

The curve can also be described by the implicit equation $g(x, y)$, such that the inverse image:

$$g^{-1}(0) = \{(x, y) \in \mathbb{R}^2 \ : \ g(x, y) = 0\}$$

In other words, The set of points (x, y) of the curve satisfy $g(x, y) = 0$. The function g classifies the points in the plane with respect to the curve.

Note that the implicit form defines the constraints which points of the ambient space must undergo in order to belong to the media object. We will exploit this for the Neural Implicit formulation.

3.2 Neural Networks

A deep neural network is a feedforward neural network with multiple layers between the input and output layers.

In a feedforward network, the computation flows from the input nodes, through the hidden nodes and to the output nodes.

There are different types of neural networks but they consist of the same basic components: neurons, weights, and activation functions. These components are inspired by the human brain functionality. The network can be trained using a Machine Learning algorithm.

In the above sense, the network is similar to a mathematical function. Also, the network is trained with data to learn this function.

Deep architectures include variants that exploit different approaches. We are particularly interested in the multilayer perceptron (MLP) and the convolutional neural network (CNN) architectures.

A multilayer perceptron (MLP) consists of at least three fully-connected layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. Usually an MLP utilizes a supervised learning technique called backpropagation for training. Arguably, the MLP is one of the most general neural network architecture.

A convolutional neural network (CNN, or ConvNet) is a class of neural network commonly applied to image analysis. In a convolutional neural network, the hidden layers include layers that perform convolutions. The convolution layer generates a feature map, which is the input of the next layer. There other hidden layers such as pooling layers, fully connected layers, and normalization layers.

We remark that the convolutional neural network is particularly suited to media objects described in parametric form, while the multilayer perceptron network is suited to media objects described in implicit form, as we will see next.

3.3 Representation Modalities and Sampling

The functional description form of the signal representation modalities presented in subsection 3.1 has great relevance in the way signals are sampled in each modality.

In the parametric form , because the domain of $f(t)$ is intrinsic, it is natural to sample it by means of an uniform sampling grid

$$\Delta_T = \{t_i \in U | t_i = i\Delta, i \in \mathbb{Z}\}.$$

Moreover, because the sampling points $\{t_i\}$ are regularly spaced, there is no need to explicitly encode them. Therefore, the sampling set is just an array function values $\{f(t_i) = p_i\}$, assuming the sampling grid Δ_T. So the sampled representation of f is given by $\tilde{f} = \{p_1, p_2, \ldots, p_N\}$.

In the implicit form, because the domain of $g(x, y)$ is extrinsic (i.e., the ambient space), the sampling is done indirectly and not necessarily using a regular structure. So, it is necessary to specify the coordinates of the sampling points p_i where g is evaluated. In this way, the sampled representation of g is given by $\tilde{g} = \{f(p_i) = c_i\}_{i=1,\ldots,N}$, i.e., the array of pairs point / value, $\tilde{g} = \{(p_1, c_1), (p_2, c_2), \ldots, (p_N, c_n)\}$.

3.4 Differentiable Neural Implicits

A Differentiable Neural Implicit models a class of functions Φ of the form

$$F(x, \Phi, \nabla_x \Phi, \nabla_x^2 \Phi, \ldots) = 0, \quad \Phi : x \mapsto \Phi(x) \tag{16}$$

This implicit formulation takes as input spatial or spatio-temporal coordinates $x \in \mathbb{R}^m$ and, optionally, derivatives with respect to these coordinates.

The representation employs a neural network that learns the function Φ while satisfying the constraint expressed by equation 16 [2].

The neural network is a deep fully-connected MLP with a smooth activation function β:

$$\Phi(x) = W_n(\phi_{n-1} \circ \phi_{n-2} \circ \cdots \circ \phi_0) + b_n, \quad x_i \mapsto \phi_i(x_i) = \beta(W_i x_i + b_i). \tag{17}$$

where the layers of the network are $\phi_i : \mathbb{R}_i^M \mapsto \mathbb{R}_i^N$ for $i = 0, n-1$, W_i and b_i are respectively the weight matrix and bias vector of the MLP that are applied to the layer input x_i and subjected to the activation function β.

The input of the network is a dataset provided in a sampled implicit format $\mathcal{D} = \{(x_i, a_i(x))\}_i$.

In order to learn the function Φ we must solve the feasibility problem that satisfies the set of constraints $\{\mathcal{C}_m(a(x), \Phi(x), \nabla\Phi(x), \ldots)\}_{m=1}^M$, i.e.:

$$\text{find } \Phi(x) \quad \text{subject to } \mathcal{C}_m(a(x), \Phi(x), \nabla\Phi(x), \ldots) = 0, \quad \forall x \in \Omega_m \tag{18}$$

Therefore, the network is trained using the following loss function

$$\mathcal{L} = \int_\Omega \sum_{m=1}^M 1_{\Omega_m}(x) \, \|\mathcal{C}_m(a(x), \Phi(x), \nabla\Phi(x), \ldots)\| \, dx \tag{19}$$

where $1_{\Omega_m}(x)$ is the indicator function of the set Ω_m.

Since in this representation Φ is defined on the continuous domain of x, it is more memory efficient than a discrete representation. Furthermore, being differentiable implies that gradients and higher-order derivatives can be computed analytically.

In sum, differentiable neural implicits constitute a suitable and powerful representation for media objects such as sound, images, videos and also for the learning 3D shapes (described as signed distance functions).

3.5 Spectral NI Architectures

Spectral Neural Implicit Architectures constitute a particular form of Neural Implicit representation in which the non-linear activation function is the periodic function $\sin(x)$. One of the main advantages of this kind of network is that it allows us to bridge the gap between the spatial and spectral domains given the close relationship of $\sin(x)$ with the Fourier basis.

3.5.1 Sinusoidal Representation Network

The sinusoidal representation network (SIREN) was proposed by Sitzmann et al. [2] as network architecture for implicit signal representation. It is an MLP model, as described in the previous section, that uses sinusoidal activation functions.

Neural representations with periodic activation functions, such as $\sin(x)$, have been regarded as difficult to train [3]. One of the key contributions of this work is the initialization scheme for training the representation that guarantees stability and good convergence. Furthermore, it also allows modeling fine details in accordance with the signal's frequency content.

Figure 2 illustrates the SIREN network architecture.

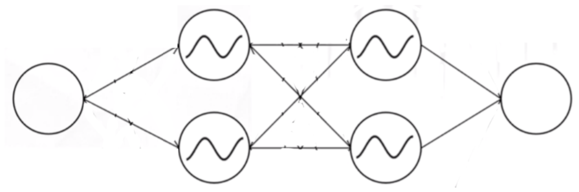

Figure 2: SIREN Network

3.5.2 Multiplicative Filter Network

A multiplicative filter network (MFN) is a spectral neural implicit architecture simpler than SIREN. In a sense it is equivalent to a shallow neural network using a filter as the nonlinear element.

The MFN repeatedly applies nonlinear filters (such as a sinusoid) to the network's input, then multiplies together linear functions of these resulting features.

The FourierNet uses a simple sinusoidal filter

$$g_i(x, \theta_i) = \sin(\omega_i x + \phi_i)$$

with parameters $\theta_i = \{\omega_i, \phi_i\}$, respectively frequency and phase .

In a forwarding pass through the network, the input coordinate x is first passed through filter layers $g_i(x)$ and then through intermediate linear layers using the following recursion

$$\begin{aligned} z_0 &= g_0(x) \\ z_i &= g_i(x) \circ (W_i z_{i-1} + b_i), \quad 0 \le i \le N_L \\ y_i &= W_i^{\text{out}} z_i + b_i^{\text{out}} \end{aligned} \tag{20}$$

where $\circ$ denotes the Hadammard product and N_L is the number of layers in the network.

It can be shown that the output of a Fourier Network is given by a linear combination of sinusoidal basis

$$y_i = \sum_{j=0}^{N_{\text{sine}}^{(i)}-1} \alpha_j \sin(\omega_j x + \phi_j) \tag{21}$$

This is due to the trigonometric identity that

$$\sin(a)\sin(b) = \frac{1}{2}(\sin(a+b-\pi/2) + \sin(a-b+\pi/2))$$

As a consequence of the multiplicative properties of Fourier filters, the entire function is ultimately just a linear function of (an exponential number of) these Fourier features of the input. Therefore, the Multiplicative Filter Network can be used to control the frequency content of the reconstructed signal.

In [4], a network architecture based on the MFN is presented. It is called BACON (Band-limited Coordinate Network). The proposed architecture produces intermediate outputs with an analytical spectral bandwidth that can be specified at initialization and achieves multi-resolution decomposition of the output.

Figure 3 shows a diagram of the Band-limited Coordinate Network.

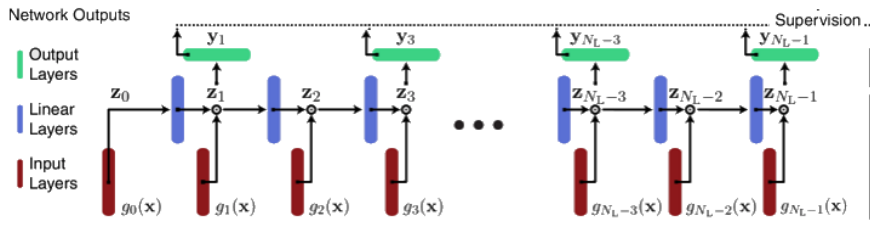

Figure 3: BACON Architecture

3.6 Filtering and Level of Detail

.

As we have seen in the previous subsection, a shallow network with just one layer composed of sinusoidal activation functions can filter a signal by band-limiting its frequency content. Thus, it provides a tool for controlling the level of detail of the output signal.

This conclusion is natural since the resulting signal representation is a linear combination of sinusoidal functions with induced frequency band. In a sense, the network is projecting the input signal into a learned dictionary of spectral atoms.

In that context, a one-layer SIREN network could also be used as a spectral filter, just as in the MFN-based architectures.

To verify the hypothesis above we conducted two simple experiments where we trained a shallow SIREN controlling the initialization of the weights of the first linear transformation, and observed how this can determine the final reconstruction.

Since the risk landscape for the SIREN loss has many local minima, is expected that these weights can't move much further than their initialization, therefore controlling where the weights start should control the range of frequencies one can filter in from the ground truth wave.

In the fist experiment, the input signal is a combination of four tones, two with lower frequencies (1Hz and 5Hz) and two more with higher frequencies (40Hz and 50Hz).

The hyper-parameters of the network and training are: sample size: 1000; hidden layers: 0; total steps: 300; learning rate: 1e-2; optimization method: Adam; number of neurons: 100.

We first tried to recover both low frequencies together. For w0 = 10 the expected behavior seems to hold, the predicted wave appears to approximate only the low frequency portion of the signal. Next we change the initialization to frequencies between 45 and 55 Hz, that also appears to succeed in predicting the high frequency tone.

See Figure 4 for the results.

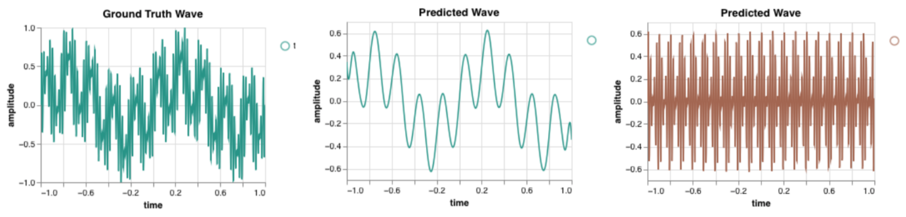

Figure 4: SIREN with one layer

In the second experiment we observe that this "decomposition" property seems not possible to obtain for deep networks, i.e., with more than one layer. We consider the sum of just two harmonics, a low frequency one with 1 Hz and a high frequency one of 50 Hz and tried the decomposition using a SIREN network having one hidden layer (so two layers in total).

We tried as before to recover the low frequency. Here we set w0 to 2, but in this case we failed to recover only the low frequency. The network despite being initialized near the first tone, it seems to be able to recover even the higher frequency (In fact with more steps and neurons it does approximates quite well).

See Figure 5 for the results.

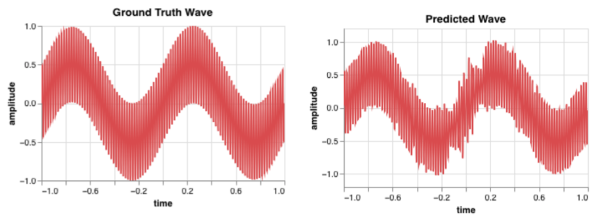

Figure 5: SIREN with two layers

Surprisingly, initializing the weights in a close range [49, 51] does seems to be able to isolate the high frequency. On the other hand if we loosen up this range a bit, to [45, 55], then the network ends up recovering the whole thing.

See Figure 6 for the results.

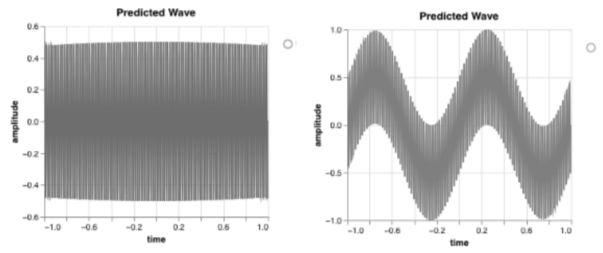

Figure 6: SIREN with two layers

Therefore it seems to be that, in general, it not straightforward to decompose the frequencies of a signal using a sinusoidal deep network with more than one layer, except for special cases where the signal is simple and the initialization is in a range very close to the target.

We will try to reason about these experiments in the next subsection.

3.7 Back to Basis, Frames and Atoms

As discussed in Section 1.1, we can define a representation operator using a family of functions, such as, bases, frames and dictionaries. We also remarked that these functions can be associated with the spatial and frequency domains, as well as with the scale and detail domains.

The properties of these representations are connected to the localization of these functions in the different domains. In that respect, it is enlightening to see the graph of these functions and visualize their characteristics.

In particular, we will look at the spectral, scale and detail basis. Regarding the localization property, these basis functions can be local or global.

The sinusoidal function $\sin(x)$ is global, since it has an infinite support. It oscillates and, as such, configures a spectral basis that detects frequencies present in the entire signal. Figure 7 shows the graph of $\sin(x)$.

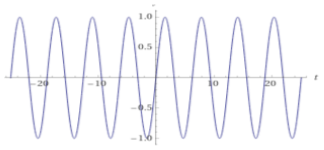

Figure 7: Graph of the Sine function

The Gaussian function is the prototypical basis of scale spaces. It has good joint localization in space / frequency and can be used to generate approximations of a signal in multi-resolution. See Figure 8.

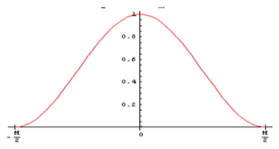

Figure 8: Gaussian Function

In order to detect local variations of a signal we employ a localized function that oscillates, such as the Laplacian of Gassusian (LoG). Intuitively it generates spaces with the details that are introduced to change between consecutive levels in a multi-resolution. This interpretation is corroborated by noticing that the LoG is approximated by difference of Gaussians with variances corresponding to two consecutive scales. This basis function is also known as the Mexican hat wavelet. See Figure 9.

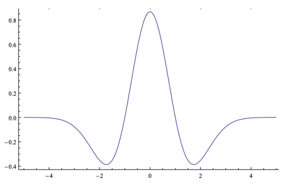

Figure 9: Difference of Gaussians - Laplacian

All the functions presented above are basis of classical spaces (i.e., spectral, scale and detail). These constitute *linear* signal representations in the sense that they do not depend on the specific content of individual signals.

On the other hand, when we employ a neural implicit representation, such as SIREN or MFN, the network is trained to reconstruct a signal from its samples by a combination of atoms that are learned to fit the particular signal. For that reason, the representation is called *non-linear*.

In subsection3.6, the first experiment used a SIREN with just one layer. Thus, it was equivalent to projecting the signal onto a dictionary of sinusoidal atoms. It was natural that could filter specific signal frequencies in the reconstruction.

However, the second experiment used a SIREN with two layers. In this case, the signal reconstruction corresponds to a learned composition of sinusoidal functions, i.e. the reconstructed signal given by a dictionary of (learned) atoms of the form $\sin \circ \sin \circ \cdots \circ \sin$. This fact has two important consequences: i) the capacity of a SIREN deep network is greater than the capacity of a shallow one with the same number of neurons; ii) for the deep SIREN network the initialization alone does not control the frequency bandwidth of the output signal as in the case of the shallow SIREN network.

In terms of localization properties in space and frequency, we can say the deep SIREN atoms 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 Figure 10 that show the graph of $\sin(3\sin(5\sin(1.9x)))$.

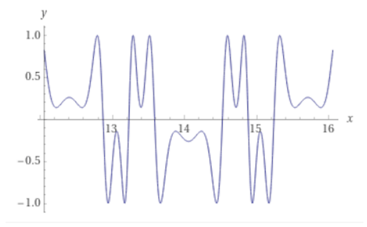

Figure 10: Atom for SIREN with 3 layers.

The conclusion, is that if we want to employ a deep SIREN network to create a signal representation with multiple levels of detail, such as in a multi-resolution decomposition, we need to find an alternative approach involving more than just the network initialization. This will be the theme of the next section.

4 Multiresolution Sinusoidal Neural Networks

In this section we propose a new approach for the representation of signals in multiple levels of detail using deep neural networks.

4.1 Overview

Our proposal is a family of coordinate-based neural networks with unified architecture that we call *MR-Net* (Multiresolution Sinusoidal Neural Networks). We derive three main variants, namely *S-Net*, *L-Net*, and *M-Net*. As a whole, they provide different trade-offs with respect to control of frequencies in the representation.

The characteristics of the MR-Net Family are:

- 3 Types of Level of Detail – in this respect it can be based on network capacity or spectral projection.
- 3 Types of Sampling – the input signal can be given either by regular sampling (with or without subsampling) or by stochastic sampling.
- Progressive Training – the network is trained progressively 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 level of detail.

The following subsections present the concepts involved in our approach, as well as, the technical details of the MR-Net architecture.

4.2 Learning Frequencies

As we have seen in Section 3.6, a neural network that employs periodic activation functions, such as $\sin(x)$, learns the signal in the frequency domain. This fact is the key to control frequencies using the network.

4.2.1 Spectral Networks

Sinusoidal networks constitute a special class of neural network that operate on a spectral representation of the signal.

The first layer of the network can be interpreted as a projection of the input signal (given by samples in the Shannon basis), to a dictionary of Sine basis.

The hidden layers of the network is constituted by spectral atoms that learn correlations of underlying frequencies of the signal.

The output layer of the network reconstructs the signal as the linear combination of a composition of spectral atoms in the last hidden layer.

Two examples of spectral sinusoidal networks are: SINE-Net [3] and SIREN (Sinusoidal Implicit Representation Network) [2].

Neural networks using periodic activation functions traditionally have been considered unstable and difficult to train and, for this reason, have not been adopted for machine learning until recently.

This problem was studied in [3] that related it with the periodicity of the sine function which implied in many local minima of the Loss landscape. However, the authors concluded that a careful initialization of the network close to the desired minima would provide stability. Further, in [2], a clever initialization scheme was proposed, which solved the problem and motivated a wide adoption of this class of network by the research community.

The main theorem of this work, provides an initialization scheme necessary for effective training based on conditions that preserve the distribution of activations through the network irrespective the number of layers. Furthermore, it introduces a factor ω_0 that increases the spatial frequency of the first layer to better match the frequency spectrum of the signal. Additionally, it uses a factorization of the weight matrix to incorporate ω_0 in all layers of the network in order to boost the gradients of the network.

We are going to adopt the sinusoidal network and initialization scheme described above, with the difference that the ω_0 factor will be used only for the initialization of the first layer and we create an additional factor, ω_G that is applied to the other layers to boost the gradients. Below, we make an analysis of the influence of these two factors in the learning process, using as example the simplest possible network.

Let $f_\theta : \mathbb{R} \to \mathbb{R}$ be a network defined by $f(x;\theta) = c \sin\big(\omega_G b \sin(\omega_0 a x)\big)$, where $\theta = (a, b, c) \in \mathbb{R}^3$ are the *unknown* network parameters and ω_0 and ω_G are known positive numbers. For simplicity, no bias was used.

Let $g : \mathbb{R} \to \mathbb{R}$ be a function which we would like to approximate by f_θ. For this, we can define a loss function $\mathcal{L}(\theta) = \int \big(g(x) - f(x,\theta)\big)^2 dx$ and use the *gradient descent* to approximate a minimum of $\mathcal{L}$ by iterating:

$$\theta_{i+1} = \theta_i - r\nabla\mathcal{L}(\theta_i) = \theta_i + 2r\int \big(g(x) - f(x,\theta_i)\big)\nabla_\theta f(x;\theta_i)\, dx$$

Where $r > 0$ is the *learning rate*. We can compute $\nabla_\theta f(x;\theta_i)$ explicitly

$$\nabla_\theta f(x;\theta_i) = \begin{bmatrix} c_i \cos\left(\omega_G b_i \sin(\omega_0 a_i x)\right) \omega_G b_i \cos(\omega_0 a_i x)\omega_0 x \\ c_i \cos\left(\omega_G b_i \sin(\omega_0 a_i x)\right) \omega_G \sin(\omega_0 a_i x) \\ \sin\left(\omega_G b_i \sin(\omega_0 a_i x)\right) \end{bmatrix}$$

This leads us to

$$a_{i+1} = a_i + 2r\omega_0\omega_G b_i c_i \int \big(g(x) - f(x,\theta)\big) \cos\big(\omega_G b_i \sin(\omega_0 a_i x)\big) \cos(\omega_0 a_i x) x\, dx$$

$$b_{i+1} = b_i + 2r\omega_G c_i \int \big(g(x) - f(x,\theta)\big) \cos\big(\omega_G b_i \sin(\omega_0 a_i x)\big) \sin(\omega_0 a_i x)\, dx$$

$$c_{i+1} = c_i + 2r \int \big(g(x) - f(x,\theta)\big) \sin\left(\omega_G b_i \sin(\omega_0 a_i x)\right)\, dx$$

The *steps* for a_i are multiplied by $\omega_0\omega_G$, and b_i by ω_G. Thus, if $\omega_G > 1$, we are speeding up the steps of a_i and b_i. In other words, more importance are given to them. see Appendix for more details on this scaling argument.

4.2.2 Frequency Initialization

The frequency initialization of the first layer, which depends on the ω_0 factor, is of central importance in the convergence of network training and multi-resolution learning.

Ideally we should initialize the network with frequencies that match the frequency content of the target signal. However, as we don't have access to this information in many applications, we must guess a value for ω_0. In [2], the authors claim to have used a fixed value $\omega_0 = 30$ which worked well for most experiments. We evaluated the impact of training a model to reconstruct one-dimensional signals using different ω_0 in initialization.

For these experiments, we used a deep SIREN network with 3 hidden layers and 256 units per layer. Our target signal was a Perlin Noise, a procedural generated signal with stochastic nature. We generated some signals with highly detailed structures by summing 16 octaves, as we call each layer of random contributions in the noise algorithm. This way, the maximum frequency of the target signal was limited only by its sampling rate. We sampled 1024 points uniformly over the interval [-1, 1], which gives a sampling rate of 512 Hz. By the Shannon-Whitaker sampling theorem, we can't represent frequencies higher than 256 (half the sampling rate). Figure 11 shows the Fast Fourier Transform of our signal, which presents a distribution of multiple frequencies up to 256.

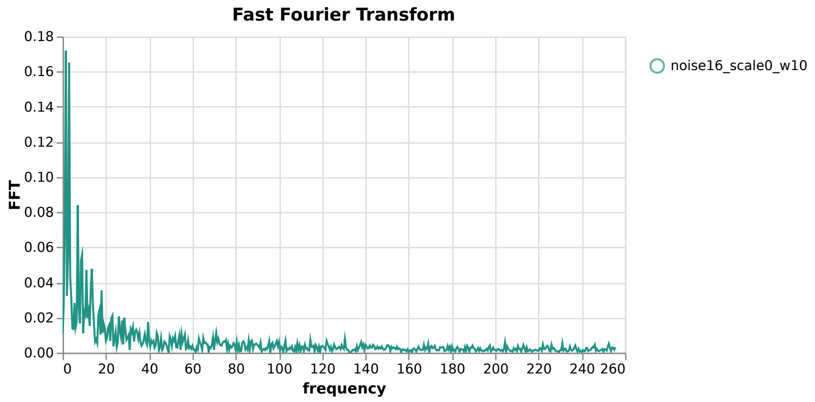

Figure 11: Fast Fourier Transform of target signal

We experimented different values of ω_0 between 10 and 1000. For signals with the characteristics described, values between 200 and 400 resulted in the best approximations by the neural network. We trained the network for only 200 epochs and by around epoch 50 we had already a reasonable result with the loss value (10^{-2}) very close to the final loss (10^{-3}).

Figure 12 shows the L2 loss evolution over training and the original and the reconstructed signals superposed on the same plot.

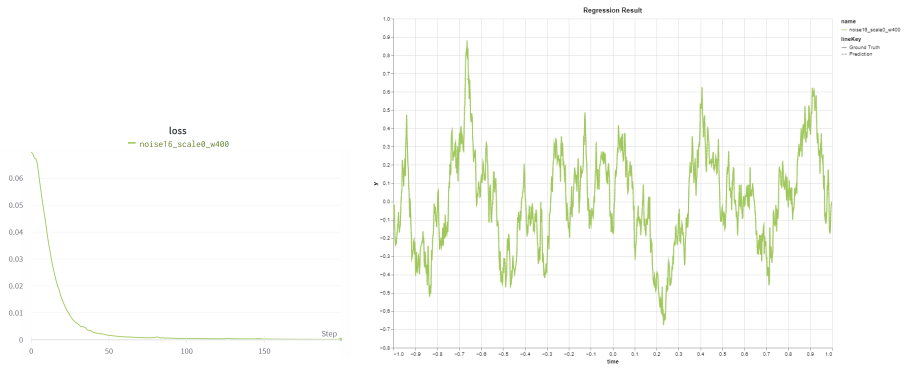

Figure 12: Regression of the signal using Siren network and $\omega_0 = 400$

Note that on this scale the signals look perfectly superposed. Figure 13 shows two detail views of the same reconstructions as it's only possible to see some difference when we look very closely.

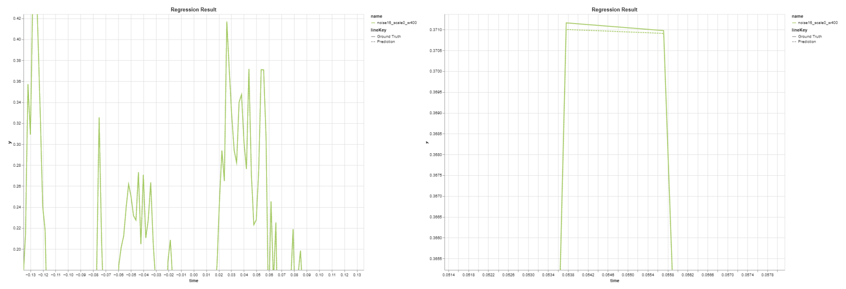

Figure 13: Detail view of reconstruction using $\omega_0 = 400$

The most interesting results, however, are observed when we initialize the network with values of ω_0 that are too large or too small for the target signal. For instance, when training the network using $\omega_0 \geq 600$, the network diverges to a noise with no apparent relationship to the target signal. Figure 14 shows the loss function value, which increases after 50 epochs, and the result prediction of the network for this case.

If we use a very small ω_0, such as 10, the network learns a filtered version of the original signal as displayed in Figure 15. By comparing the Fast Fourier Transform of the learned signal and the target signal, we concluded that the network learns the low frequencies of the signal up to an unknown value that increases or decreases with ω_0.

Figure 16 shows the spectrum of frequencies of the target signal and its reconstruction using $\omega_0 = 10$. Notice how the Fast Fourier Transform of the reconstructed signal matches that of the original signal up to a value around 12, then it fails to capture the exact contribution of frequencies between 12 and 50

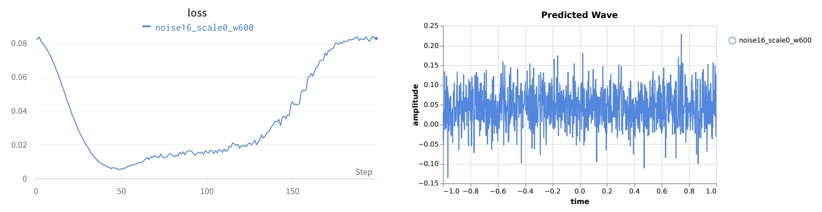

Figure 14: Network can't learn with $\omega_0 = 600$

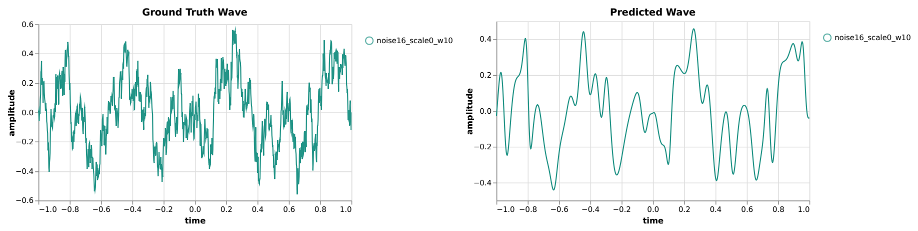

Figure 15: Result after training with $\omega_0 = 10$

and finally it does not exhibit contributions of higher frequencies.

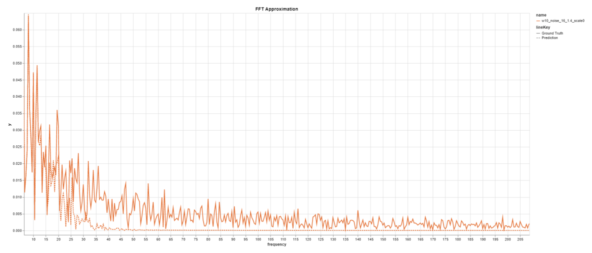

Figure 16: Network learning low frequencies due to low w0 value

4.2.3 Capacity Filtering

So far, we have worked with a high capacity Siren Network using 3 hidden layers with 256 units per layer. As we saw, this network can approximate our 1D signals perfectly and, in fact, we know it can approximate much more complex signals. For example, representations of neural implicit surfaces in three and higher dimensions [5], [6].

We decreased the number of layers and computational units to evaluate what happens when we try to approximate a signal using a SIREN model with lower capacity than it would be necessary. Experiments showed us that we can also reconstruct a filtered version of the target signal when using low capacity SIREN models initialized with small ω_0.

4.2.4 Frequency / Detail Splitting

As we have seen in the experiments described in the two previous sections there are two mechanisms to control the frequency band (or level of detail) learned by the sine network from a signal. They are: i) the initialization of weights (frequencies) of the first network layer; and ii) the capacity (number of neurons and layers) of the network. (Note that these mechanisms can be combined.)

To this end, if we want to have multiple levels of detail represented in the network, one option is to split the network in parts, such that each part encodes one level of detail. As a consequence, such an option motivated us to devise a module that could be used to partition the structure of the network.

MR-Module

Our goal is to design a module capable of approximating a signal with multiple frequency bands. When we introduce hidden layers, we know, from previous experiments, that the model capacity increases and we can't control the exact content of frequencies the network learns. This way, we aimed for a module with a single hidden layer and used Perlin Noises with 4 local extrema (Figure 17) as a perceptual indicator of ”limited frequency band”.

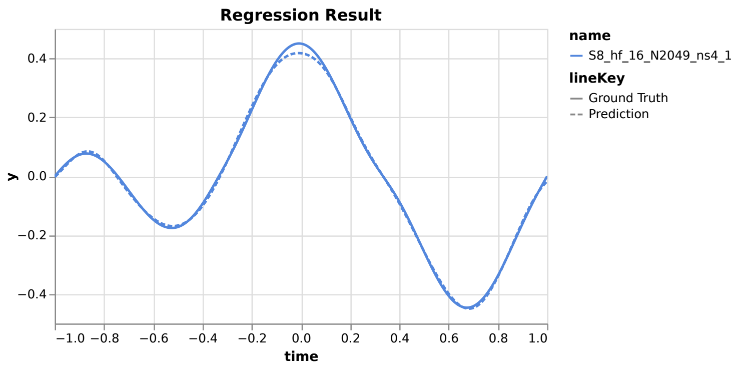

Figure 17: Reconstruction of low frequency signals using a small size model

We did experiments with 1 and 2 hidden layers modules using 8, 16 or 32 computational units in each layer. After looking at the results, we chose a module with 1 hidden layer and 16 units in each layer as our MR-Module for most experiments.

We didn't target the minimal module possible, as Figure 18 shows this module is able to approximate more complex signals up to a certain frequency. Having a small module with reasonable capacity gives us more flexibility in terms of level of detail strategy and helps to keep each stage of different MR-Net architectures uniformly as we'll discuss on the next section. However, the breadth and depth of the MR-Module could be adapted to specific applications without compromising compatibility with the MR-Nets architectures.

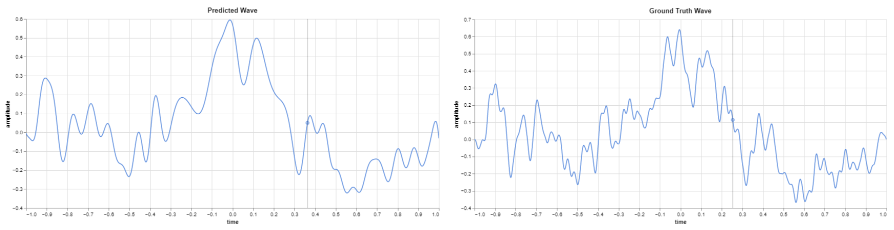

Figure 18: MR module prediction after training on a complex signal above its capacity

4.3 Architecture

Based on the considerations for learning and splitting frequencies of the input signal into levels of detail, we devised a general architecture for a family of neural networks – the Multiresolution Sinusoidal Neural Network.

The core idea is to structure the network into multiple *stages*. Each stage learns in a controlled way a level of detail corresponding to a frequency band.

A stage has a configuration derived from the MR-Module described in the previous section and they are trained based on a predefined schedule (See Section 4.4).

Figure 19 depicts the anatomy of the network, where we can see the N stages of MR-Modules, each composed of four blocks of layers: the first layer; the hidden layers; the linear layer; and the control layer.

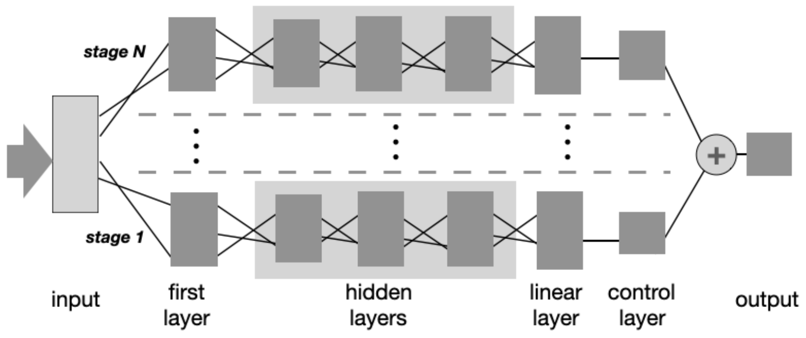

Figure 19: Anatomy of MR-Net Family

The three initial layers form a fully connected MLP. The last layer consists of one node (not trained) to control the stage output, given by the function $c_\alpha(x) = \alpha x$, with $\alpha \in [0, 1]$ and x comes from the linear layer. During training, the control layer is kept constant $\alpha = 1$, i.e., equal the identity.

We can say that the first layer performs a projection of the input signal into a dictionary of sine functions, the hidden layers correspond to correlations of order

n of signal frequencies, the linear layer reconstructs the signal as a combination of these frequency atoms, and the control layer is just a mechanism to provide a continuous blend of level-of-detail in the network.

The contribution of these N stages is added together forming the network output. Assuming that the MR-Net is learning a function $f(x)$ that fits the input signal, then

$$f(x) = g_1(x) + \cdots + g_N(x) \tag{22}$$

where $g_i(x)$ is the detail function given by the stage s_i, for $i = [1, N]$. The first stage s_1 corresponds to the coarsest approximation of the signal and the other subsequent stages add increasingly finer details to it.

Note that this architecture is very much in the spirit of the Multiresolution Analysis [7]. Indeed, consider the base case with $f(x) = g_1(x) + g_2(x)$, then $g_2(x) = f(x) - g_1(x)$, i.e., g_2 are the details that need to be added to g_1 to increase the level of detail.

The network learns the decomposition of the signal as the projection into the coarse scale space and a sequence of finer detail spaces. The characteristics of the level of detail decomposition of each member of the MR-Net family will depend of the specific configuration of the network stages, as will be presented next.

4.3.1 S-Net

In the S-Net, a stage has only two blocks (plus the control which is not involved in the learning process): one for the first layer, another for the linear layer (and the control layer). (See Figure 20.) Therefore, this kind of network consists of a learned "sine transform", with the two layers corresponding, respectively, to the direct and inverse sine transform. As a consequence, the S-Net can provide level of detail by the initialization of frequency bands in each stage.

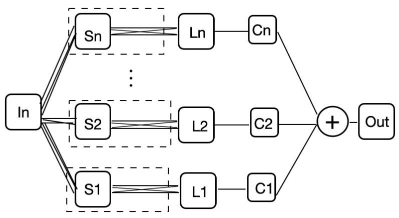

Figure 20: S-Net

4.3.2 L-Net

The L-Net is composed of N complete independent stages (with the four blocks: first; hidden; linear and control) that are aggregated in the output. (See Figure 21.) Consequently, the level of detail in this kind of network is determined by the capacity of the individual stages.

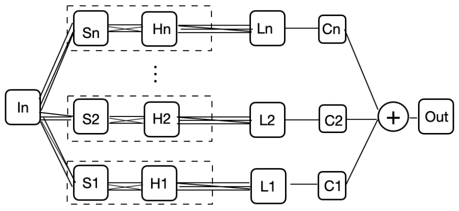

Figure 21: L-Net

4.3.3 M-Net

The M-Net consists of a hierarchy of stages, in which subsequent stages are linked together. The output of a block of hidden layers is connected both to the linear layer, as well as, to the input of the hidden layer block of the next stage. (See Figure 22.)

A consequence of this hierarchical structure is that the hidden layer block in a stage is augmented with the sequence of hidden layer blocks coming from previous stages. Therefore, the capacity of each stage increases with its depth in the hierarchy. Accordingly, it is expected that this kind of network provides a more powerful mechanism for learning levels of detail.

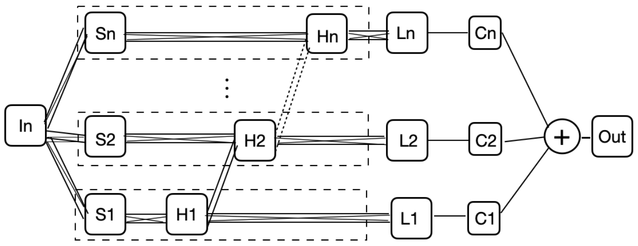

Figure 22: M-Net

4.4 Training

The training of the MR-Net has to take into account the mechanisms for learning different levels of detail by each stage of the network. There are two basic mechanisms: i) level of detail filtering and ii) pre-processing of the input signal.

As we have discussed in Section 4.2.4, level of detail filtering is achieve either with the frequency band filtering (through the initialization of the w_0 weights of the first layer); or with the capacity filtering (conditioned on the number of nodes and layers of the network).

Regarding the network input, we can either use the original signal, or pre-process the signal with a low-pass filter.

4.4.1 Multi-Stage Schedule

Since the M-Net has multiple stages, each learning a different level of detail, one important aspect is the stage training schedule. We could train the whole network with all stages in parallel, but it might be beneficial to train each stage in sequence from the lowest to the highest level of detail. This scheduling is our choice and a common strategy in the traditional multiresolution analysis of signals.

4.4.2 Progressive learning

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

4.4.3 Adaptive Training

We also employ an adaptive training scheme for the optimization of each network stage combining both accuracy loss thresholds and convergence rates.

4.5 Sampling

As we have discussed in Section 2, Sampling is a way to create a representation of a function based on its values at certain points of the domain. From the point of view of Representation Theory (Section 1.1), we can interpret this as a projection of the function 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 of signal values at the grid locations. (See Figure 23.)

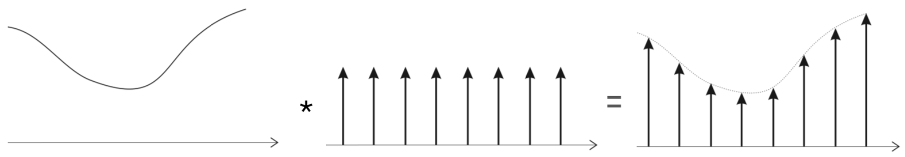

Figure 23: Sampling a signal using a train of impulses.

When we employ a coordinate-based neural network to learn a signal, we provide the network with the pairs $\{(x_i, v_i)\}_{i=1,N}$, where x_i are the coordinates of the sampling grid locations and the $v_i = f(x_i)$ are the signal values at those locations.

One important aspect of sampling for learning signals with this kind of neural network is the structure of the sampling grid. In that respect, it is instrumental to consider four types of grid structures: Regular, Irregular; Multiresolution and Stratified.

4.5.1 Regular

In a regular grid, the sampling points are uniformly spaced, forming a regular pattern. (See Figure 24.)

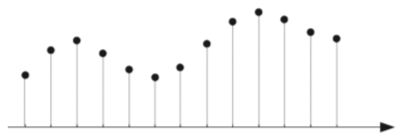

Figure 24: Regular Sampling

4.5.2 Irregular

In an irregular grid, the sampling points have a random placement, forming an irregular pattern. (See Figure 25.)

Two common strategies to construct irregular grids are: jittering and stochasticity. Jittering starts with a regular grid and the sampling points are randomly displaced. In a stochastic construction the sampling points are placed according to a random distribution, such the Poisson Disk distribution.

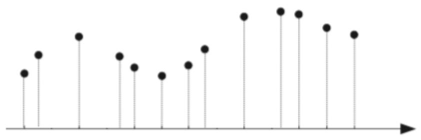

Figure 25: Stochastic Sampling

4.5.3 Multiresolution

A Multiresolution grid consists of a hierarchy of sampling grids, with different resolutions. The standard grid structure form a dyadic lattice of regular grids in which the resolutions obey the 2^j rule – i.e., each level of the grid has twice the size of the previous one. (See Figure 26.)

Nonetheless, it is also possible to define a irregular multiresolution grid structure. In this case, each resolution level has approximately twice the number of random sample points of the previous level.

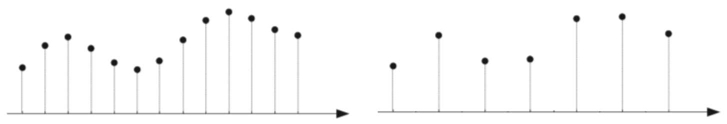

Figure 26: Multiresolution Sampling

4.5.4 Stratified

In a stratified grid, the sample points are allocated according to their importance. The structure consists of a collection of strata defined by the allocation criteria.

For example, we could choose to sample a function at its extrema (i.e., the set of local maxima and minima). Figure 27 illustrates this example (the vectors indicate the tangent of the curve).

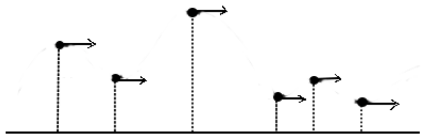

Figure 27: Stratified Sampling

4.6 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 the family of Multiresolution Sinusoidal Neural Networks. The main ones are: Capacity Based with Original Signal; Filtering with Gaussian Tower; and Filtering with Gaussian Pyramid.

4.6.1 Capacity Based

In this level of detail scheme, we train each stage of a MR-Net using all samples of the original signal. The first stage is initialized with the lowest value for ω_0 and learns the lowest frequencies of the signal up to a limit given by its capacity. Then, we add another stage to the network, initialized with a higher ω_0, to learn more details of the target signal. We can add more stages until reach a desired error tolerance or a defined maximum number of stages.

Figure 28 shows the four first stages of an M-Net trained to learn a signal using the maximum capacity of details per stage. Notice as each stage adds more details to the previous one.

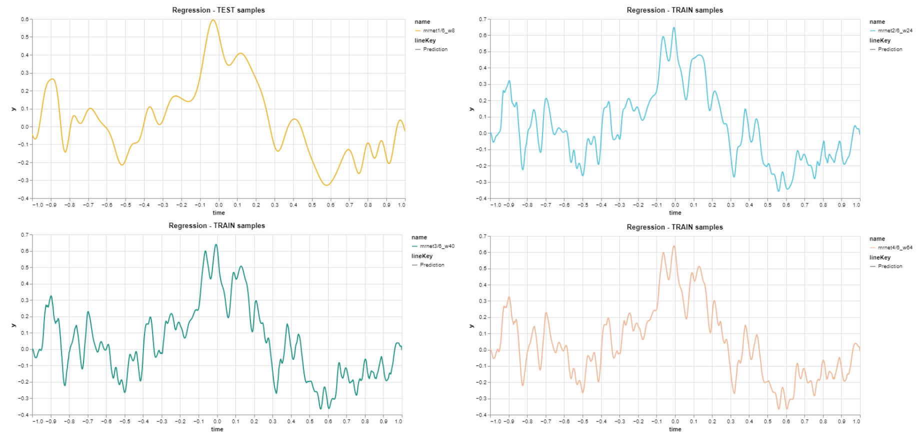

Figure 28: M-Net stages 1 to 4 trained on the same data

Figure 29 shows the ground truth signal and the final reconstruction of the M-Net after training 5 stages. The reconstruction is so accurate that the ground truth and prediction curves are overlapping.

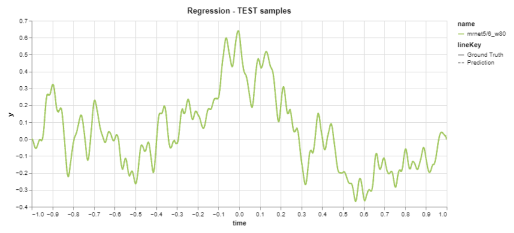

Figure 29: M-Net final reconstruction of a signal using 5 stages

4.6.2 Gaussian Tower

If we want to have control over the frequencies present in the signal we can build a multiscale representation of the signal and train the network to approximate each level of this representation.

We start by feeding our model with a ”Gaussian Tower”, that is, multiple versions of the signal filtered consecutively by a low-pass filter, but without decimation. This way, each scale is reconstructed from the same amount of samples. The network must be trained from the less detailed scale to the most detailed one.

Figure 30 shows the result of training a M-Net with an 8-level Gaussian Tower of a Perlin Noise with 2049 samples. Once again, the reconstruction is so accurate that we can't distinguish the ground truth and the model prediction on visual inspection of the whole interval (left column). We show closer views for each level in the right column to help doing qualitative evaluation of the prediction.

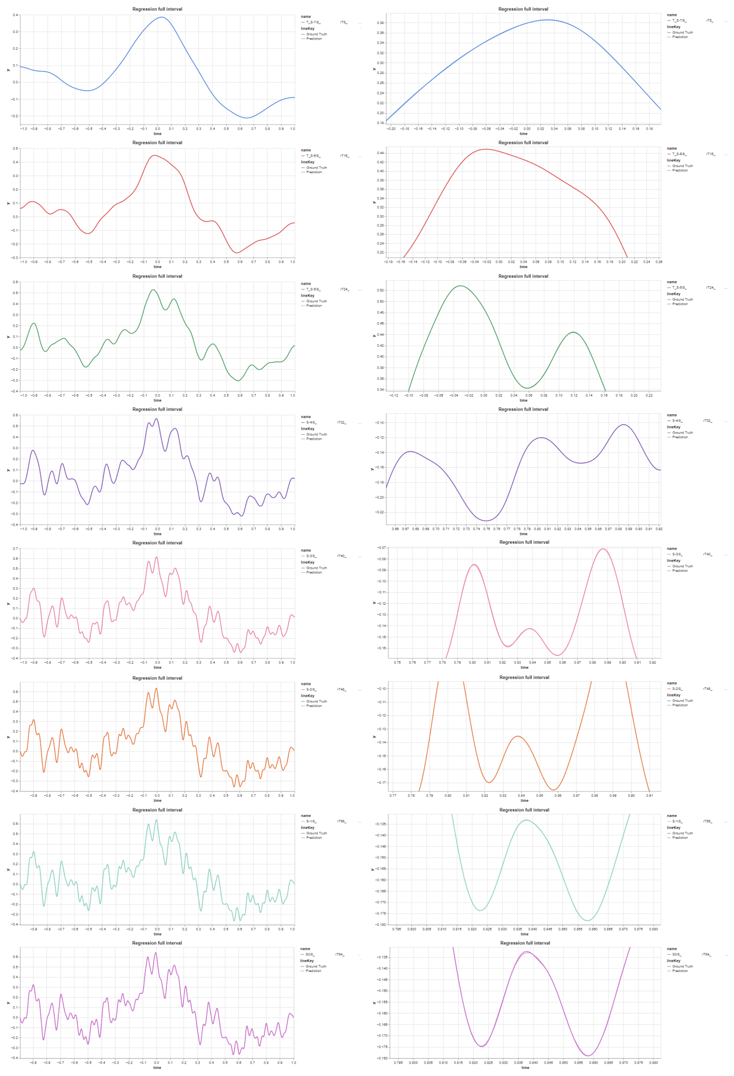

Figure 30: Gaussian Tower reconstruction using M-Net (full interval)

4.6.3 Gaussian 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.

While the reconstruction of signals using the Gaussian Tower shown in the previous subsection is perfect, it is also wasteful if we can generalize correctly the model based only on the samples of a Gaussian Pyramid. However, in our experiments with the Gaussian Pyramid, we observed high frequency oscillations appearing at intervals 'in-between' the sampled locations given in training. This effect is shown in Figure 31,

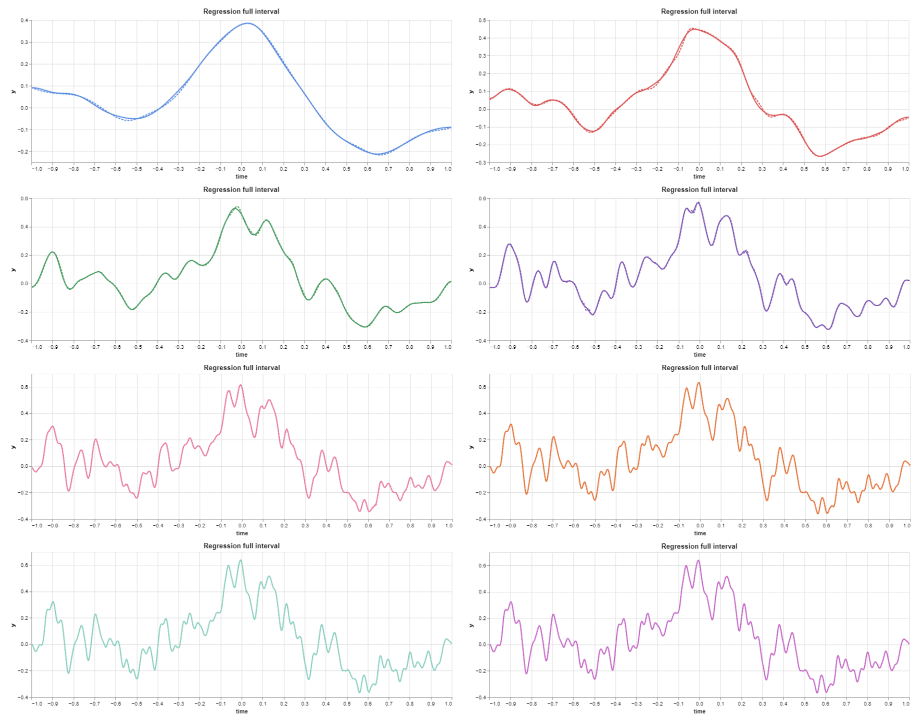

Figure 31: Gaussian Pyramid reconstruction using M-Net

These artifacts are most noticeable at the coarser levels of the pyramid, in which we use few samples for training. Figure 32 shows closer views of these levels.

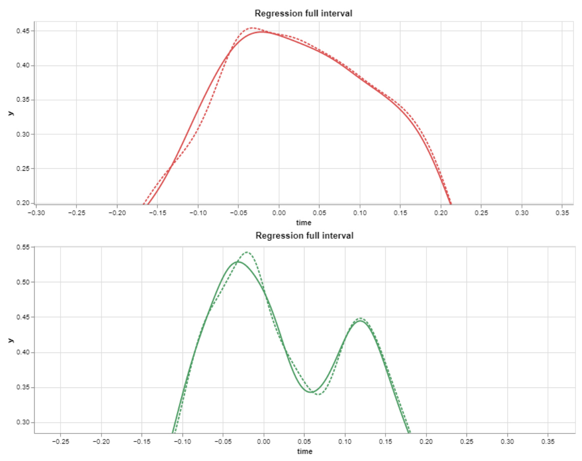

Figure 32: Detail view of 2nd and 3rd stages of Gaussian Pyramid reconstruction using M-Net

Another relevant factor in the manifestation of these artifacts is the initialization of the frequencies in each stage. Sinusoidal networks can learn higher frequencies quickly and without a regularization strategy it may exhibit spurious oscillations as those shown in Figure 33. We'll discuss this matter further in the next section.

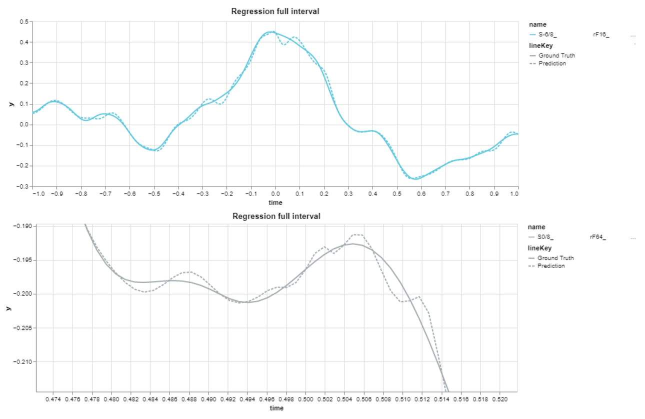

Figure 33: Gaussian Pyramid reconstruction using MR Modules with no superposition of frequencies

4.7 Loss and Regularization

The general problem we are dealing with in this work is to create a signal representation based on a set of samples with the goal to reconstruct it using a neural network.

In that context, reconstruction is related to interpolation and fitting. In other words, we would like to be able to recover the signal by: i) interpolating the sampled values; and ii) fitting a continuous function that best approximates the true signal over all points of its domain;

On the other hand, learning the representation with a neural network implies in model regression and generalization. This means that: i) the approximation should take into account a model of what we know about the class of the signal; and ii) the model should generalize the information we have at the sample points to the entire domain of the signal.

To accomplish these goals above, we use a Loss function to train the network, which measures how well our objectives have been fulfilled. Consequently, a good Loss function bridges the gap between the data and the model.

Furthermore, while training, the model can be overfitted to the data. To avoid this, regularization techniques reduce the chance of overfitting and help us to fit properly obtaining an optimal model.

4.7.1 L1 and L2 Regression

The most common Loss Functions for regression problems use the $L1$ and $L2$ norms.

These functions minimize the error which is the sum of the all the differences between the true value at the sample points $y_i = f(x_i)$ and the predicted value by the network $\hat{y}_i = \hat{f}(x_i)$.

The L1 Loss minimizes the absolute differences, i.e.,

$$L1 = \sum_{i}^{n} |y_{\text{true}} - y_{\text{pred}}|$$

while the L2 Loss minimizes the squared differences

$$L2 = \sum_{i}^{n} (y_{\text{true}} - y_{\text{pred}})^2$$

In the case of the MR-Net, if the capacity of the network does not matches well the frequency content of the signal, we can have a situation where the reconstruction interpolates the signal at the sampling points, but oscillates in intermediate intervals due to higher frequencies not present in the signal (over-capacity).

The following example (Figure 34) shows the above described phenomena.

In this case, we need to use a regularization technique, such as Frequency Bias, Weight Decay of Higher Order Data.

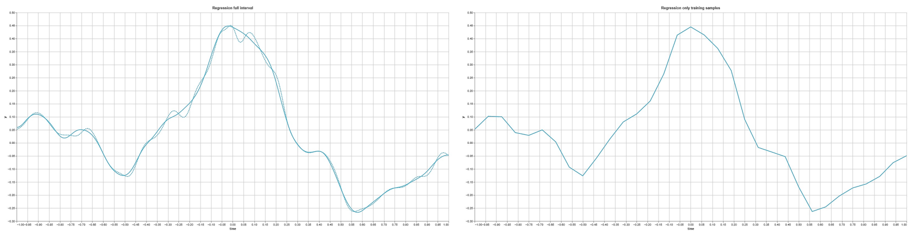

Figure 34: Network interpolates the samples, but may introduce higher frequencies in the signal

4.7.2 Frequency Bias

Frequency Bias promotes the use of lower frequencies in the initialization of the network. When initializing the first layer of the stage k, we can choose to pick random frequencies in the interval $[-\omega_{0_k}, \omega_{0_k}]$ or in the set $[-\omega_{0_k}, -\omega_{0_{k-1}}] \cup [\omega_{0_{k-1}}, \omega_{0_k}]$. That is, we can choose to only add higher frequencies than those already used in previous stages or to add frequencies with overlapping on previous intervals.

Choosing overlapping intervals acts as a frequency bias regularization, reducing the high frequency artifacts and spurious oscillations as shown in Figure 35

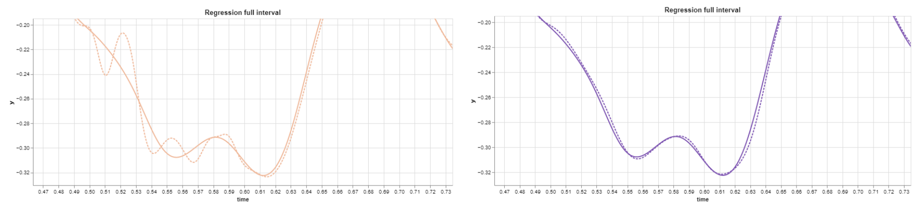

Figure 35: Regularization by frequency bias

4.7.3 Weight Decay

Weight Decay encourages lower frequencies by including a regularization them in the loss function that penalizes high frequencies (i.e., weights with large values).

4.7.4 Higher Order Data

In the normal setting the training is based on a set of values at the sampling points. Observe that this type of data gives information only at the sample grid points.

However, if we include in the data derivatives of order n of the signal, then we are giving information of the underlying function in the vicinity of the sample. This is a powerful regularization that can be incorporated into the model through an appropriate term in the Loss Function.

In the sequence, we describe experiments of using higher order data to provide better generalization of the MR-Net.

We start evaluating if this regularization strategy can help to remove or reduce the spurious oscillations observed in section 4.6.3. We initialized the stages of a M-Net with frequencies drawn from non-overlapping intervals. Then, we trained the network with a Gaussian Pyramid using the value and the 1st derivative of the signal at each sample.

Figure 36 shows the reconstruction of the 6th level of a Gaussian Pyramid. On the left, we show the result after training the M-Net using only the sample values; on the right, we show the result after training using the sample values and the 1st derivative of the signal at these positions. Note that the addition of the 1st derivative removes the spurious oscillations and makes the result smoother.

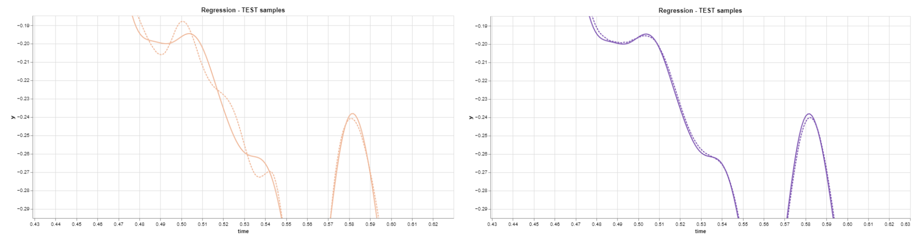

Figure 36: 6th level of the Gaussian Pyramid without and with higher order regularization

Figure 37 shows the Fast Fourier Transform of the reconstructed signals presented in Figure 36. Note how the frequencies greater than 36 were attenuated with addition of the 1st derivative.

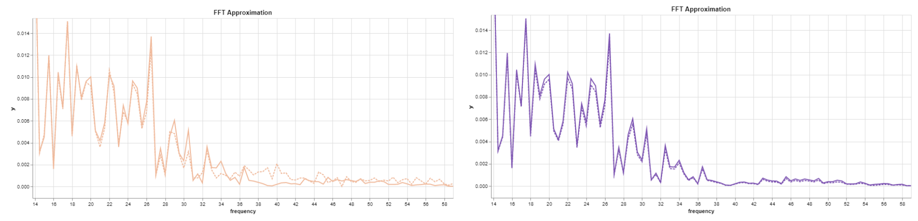

Figure 37: Fast Fourier Transform plot of the 6th level of the Gaussian Pyramid without and with higher order regularization

We can experiment multiple combinations of sampling schemes and information at sample points. A case of particular interest occurs when we are working with implicit functions and trying to reconstruct a level set, such as the 0-level set, of this function. In higher dimensions, we can use this approach to model graphical objects such as curves or surfaces. In this case, we can assume that we have higher order data at the zeros of the implicit function, our signal, and only the spatial value at other points.

We did experiments on some configurations in this direction, giving the signal value and its 1st and 2nd derivatives at its zeros plus some combinations of points given by stratified or stochastic sampling. Figure 38 shows the reconstruction of a signal using this strategy with 10 stochastic samples, on the left, and the linear interpolation of the samples used, on the right. Note that the stochastic samples aren't enough to capture some local maximum and minimum of the signal and the reconstruction fails near these points.

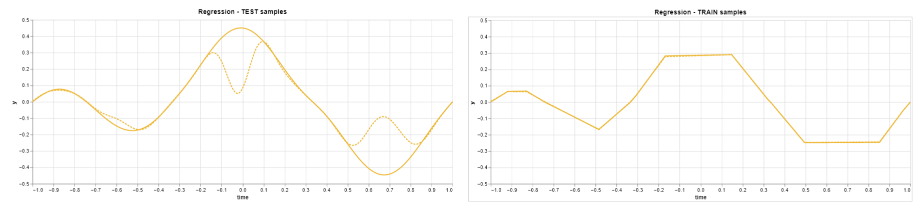

Figure 38: Signal value, 1st and 2nd derivatives at zeros plus value at 10 stochastic samples

Figure 39 shows the reconstruction using the signal value and its 1st and 2nd derivatives at its zeros plus the signal value at the extrema of the function. Notice that using feature points like the extrema values is better than using a few more random samples.

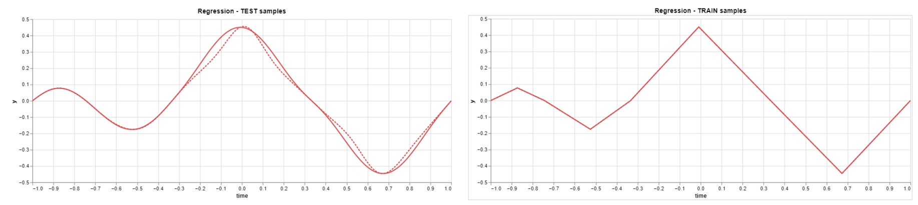

Figure 39: Signal value, 1st and 2nd derivatives at zeros plus value at extrema

If we combine the strategies by keeping the higher order data at the zeros, using feature points of the signal and adding a few more random samples, we can get a very good reconstruction as shown in Figure 40.

The first and second derivatives of the signal are also reasonably approximated (Figure 41) even though we only supervised their values at the zeros of the signal.

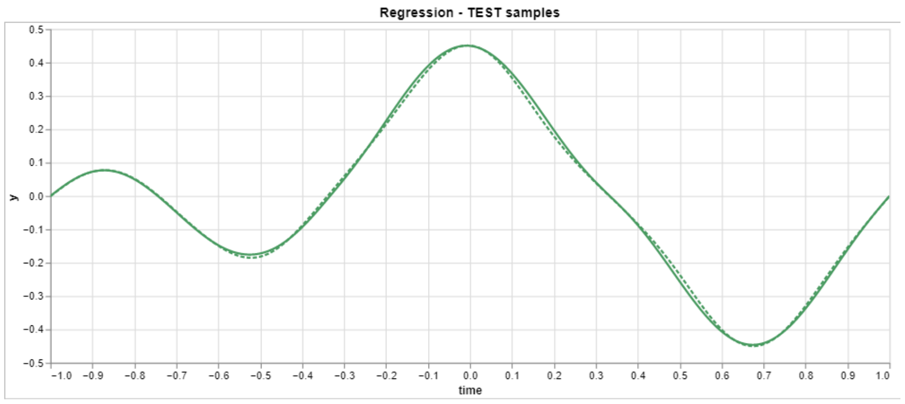

Figure 40: Signal value, 1st and 2nd derivatives at zeros plus value at extrema and 10 stochastic samples

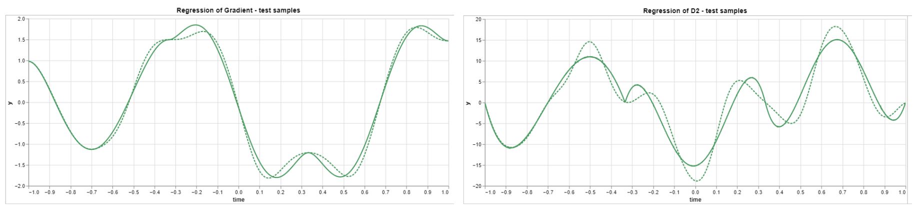

Figure 41: 1st and 2nd derivatives of the network

4.8 Reconstruction

Arguably, we can say that the primordial purpose of a representation is to provide an accurate reconstruction the underlying signal. 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 Multiresolution Sinusoidal Neural Network 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.

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

4.8.1 CPU / GPU Visualization

Another important aspect of signal reconstruction for media applications is related to visualization, which is crucial for the various distinct tasks needed in the application.

The nature of these tasks are highly application dependent. For example, Computer-Aided Design (CAD) modeling program needs to provide feedback for editing operations, while a Lighting Design App must render a realistic physically based simulation of the scene and a Game depends on real-time visual response with the best quality allowed by the hardware platform.

Our implementation of the MR-Net reconstruction (i.e., inference), is planned to support both CPU and GPU visualization. We are developing libraries in Python and C++ for CPU rendering, as well as in CUDA for GPU rendering.

For the case of Neural Implicit surfaces we are developing a ray-tracing algorithm for level-of-detail rendering [8]. In the future, we plan to make this code compatible with the GPU RTX standard.

4.8.2 Filtering / Antialiasing

As it has been demonstrated in the previous subsections, 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 at arbitrary display resolutions.

4.8.3 Progressive

Finally, the structure of our representation as 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.

References

[1] Jun Xian and Song-Hua Li. Sampling and reconstruction for shift-invariant stochastic processes. *Stochastics*, 86(1):125–134, 2014.

[2] Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Proc. NeurIPS*, 2020.

[3] Giambattista Parascandolo, Heikki Huttunen, and Tuomas Virtanen. Taming the waves: sine as activation function in deep neural networks. https://openreview.net/forum?id=Sks3zF9eg, 2017.

[4] David B. Lindell, Dave Van Veen, Jeong Joon Park, and Gordon Wetzstein. Bacon: Band-limited coordinate networks for multiscale scene representation. *arXiv preprint arXiv:0000.00000*, 2021.

[5] Tiago Novello, Vinícius da Silva, Guilherme Schardong, Luiz Schirmer, Hélio Lopes, and Luiz Velho. Differential geometry in neural implicits. *arXiv:2201.09263*, January 2022.

[6] Tiago Novello, Vinícius da Silva, Guilherme Schardong, Luiz Schirmer, Hélio Lopes, and Luiz Velho. Neural implicit surfaces in higher dimension. *arXiv:2201.09636*, January 2022.

[7] S.G. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, 1989.

[8] Vinícius da Silva, Tiago Novello, Guilherme Schardong, Luiz Schirmer, Hélio Lopes, and Luiz Velho. Mip-plicits: Level of detail factorization of neural implicits sphere tracing. *arXiv:2201.09147*, January 2022.

[9] Weights and biases. https://wandb.ai/site.

Appendix 1

Penalizing the loss function parameter

Let $\mathcal{L} : \mathbb{R} \to \mathbb{R}$ and $\widetilde{\mathcal{L}} : \mathbb{R} \to \mathbb{R}$ be two loss functions such that $\widetilde{\mathcal{L}}(\theta) = \mathcal{L}(\omega \cdot \theta)$, where $\omega \in \mathbb{R}$. Observe that $\mathcal{L}$ and $\widetilde{\mathcal{L}}$ share the same number of critical elements. More precisely, if θ is a critical element for $\mathcal{L}$, then $\frac{\theta}{\omega}$ is a critical element for $\mathcal{L}$. Given a parameter $\theta_0 \in \mathbb{R}$, the *gradient descent* approximates a minimum of $\mathcal{L}$ by iterating the system

$$\theta_{i+1} = \theta_i - r\nabla\mathcal{L}(\theta_i), \text{ with } r > 0. \tag{23}$$

On the other hand, defining $\widetilde{\theta}_0 = \frac{\theta_0}{\omega}$ and iterating the gradient descent for $\widetilde{\mathcal{L}}$ results in

$$\widetilde{\theta_{i+1}} = \widetilde{\theta}_i - \tilde{r}\nabla\widetilde{\mathcal{L}}(\widetilde{\theta}_i), \text{ with } \widetilde{r} > 0, \tag{24}$$

which is equivalent to

$$\widetilde{\theta_{i+1}} = \widetilde{\theta}_i - \tilde{r}\omega\nabla\mathcal{L}(\omega \cdot \widetilde{\theta}_i). \tag{25}$$

Proposition 3. *If $\tilde{r} = \frac{r}{\omega^2}$, then $\widetilde{\theta}_i = \frac{\theta_i}{\omega}$.*

Proof. The case $i = 0$ follows by the definition of $\widetilde{\theta}_0$. Suppose that $\widetilde{\theta_{i-1}} = \frac{\theta_{i-1}}{\omega}$, then

$$\begin{aligned}
\widetilde{\theta}_i &= \widetilde{\theta_{i-1}} - \tilde{r}\omega\nabla\mathcal{L}(\omega\widetilde{\theta_{i-1}}) \\
&= \frac{\theta_{i-1}}{\omega} - \frac{r}{\omega^2}\omega\nabla\mathcal{L}(\omega\frac{\theta_{i-1}}{\omega}) \\
&= \frac{1}{\omega}\left(\theta_{i-1} - r\nabla\mathcal{L}(\theta_{i-1})\right) = \frac{\theta_i}{\omega}.
\end{aligned}$$

□

Therefore, if we keep the same learning rate $r = \widetilde{r}$ and $\omega > 1$, the sequence of weights $\widetilde{\theta}_i$ given by the gradient descent *may* go faster towards a critical element than the sequence θ_i.

Training a network in multiresolution

Let $f : \mathbb{R} \to \mathbb{R}$ be a smooth function. We would like to approximate f by a sum of two neural networks f_{θ_1}, f_{θ_2} such that f_{θ_1} approximates the lower frequencies of f and f_{θ_2} approximates the remaining higher frequencies.

Consider the *heat equation* of f

$$\begin{cases} u_t = -k\Delta u & \text{in } \mathbb{R} \times \mathbb{R} \\ u = f & \text{on } \mathbb{R} \times \{t = 0\} \end{cases} \tag{26}$$

Recall that the function $u(x, t)$ is a "smoothing" of $f(x)$. Expanding $u(x, t)$ in a *Taylor series* near $t = 0$, we obtain:

$$u(x, t) = u(x, 0) + tu_t(x, 0) + O(t^2) \tag{27}$$

Replacing the heat equation in Equation 27 results in the following quadratic approximation of $f(x)$

$$f(x) \approx u(x, t) + tk\Delta f(x)$$

Therefore, $u(x, t)$ and $tk\Delta f(x)$ are candidates to train f_{θ_1} and f_{θ_2}. Specifically, we can define the loss functions considering:

$$\mathcal{L}_1(\theta_1) = \int_{\mathbb{R}} |u(x, t) - f_{\theta_1}(x)|\, dx \tag{28}$$

$$\mathcal{L}_2(\theta_2) = \int_{\mathbb{R}} |tk\Delta f(x) - f_{\theta_2}(x)|\, dx. \tag{29}$$

Applying induction in the above procedure gives rise to an approximation of f by a sum of n neural networks $f_{\theta_1} + \cdots + f_{\theta_n}$. This can be done by approximating $tk\Delta f(x)$ by f_{θ_n}. Then we repeat the above procedure considering $u(x, t)$ in the place of $f(x)$. After $n - 1$ iteration of this algorithm, we obtain the n desired candidates.