Sinusoidal Representation Network for Single Image Super Resolution

Angelica Urbanelli
s271114
angelica.urbanelli@studenti.polito.it

Klaus Cuko
s260351
klaus.cuko@studenti.polito.it

Roberto Bressani
s270079
roberto.bressani@studenti.polito.it

Abstract

Sinusoidal representation network (SIREN) is a recent powerful architecture that exploits implicit neural representation with periodic activation function to get a continuous representation of several kind of signals. In our report we studied this network searching for possible improvements, and then we analysed advantages and disadvantages of this architecture to apply it on single image super resolution.

1. Introduction

The starting point of our study is to build and validate an architecture able to fit different signals such as audio and images, and even to reconstruct an image starting from its gradient or laplacian by solving differential equations. In addition, we conduct an ablation study on the different components of the network to evaluate their role on the final results.
We run different experiments under various configurations in order to find the best one, and then leveraging the network representation for the super resolution task. This technique aims to obtain an high resolution (HR) image, starting from a low resolution (LR), as we will try to do with SIREN, but can also be used to restore an higher quality image from a degraded one.
We use DIV2K [1], one of the most commonly used dataset for this task, in order to compare our results with other state-of-the-art techniques, for which networks pretrained on its training dataset are available. In particular, we choose to use validation dataset[1]: `DIV2K_valid_LR_bicubic_X4` for LR, and `DIV2K_valid_HR` for HR.
For the implementation refer to `https://github.com/robertobressani/ai_ml_siren_sr`.

[1]Available at `https://data.vision.ee.ethz.ch/cvl/DIV2K/`

2. Related works

In this section, we are going to present the main concepts we based our work on. We mainly followed the work done by Sitzmann *et al.* [7] and analysed the state-of-the-art in super resolution.

Implicit Representation The goal is to learn a continuous function to represent a specific signal. This is done by training a network per each signal, so that it can be reconstructed from the representation itself. This leads to some advantages, such as finding a compressed encoding without loosing too much information and exploiting the differentiability of the function to solve additional tasks. Formally, given a signal represented as a set of pairs $(\underline{x}_i, f(\underline{x}_i))$ (where $\underline{x} \in \mathbb{R}^n$ is the coordinate and $f(\underline{x})$ its value), the objective is to find a function $\Phi : \underline{x} \to \Phi(\underline{x})$, so that Φ is the neural representation that is able to map each coordinate to a value. It can be found by solving an equation in the form:

$$F(\underline{x}, \Phi, \nabla_{\underline{x}}\Phi, \nabla^2_{\underline{x}}\Phi, ...) = 0 \tag{1}$$

To solve this problem, it can be transformed into a feasible one where Φ is the function satisfying a set of constraints that relate the function Φ itself and its derivatives to quantities $f(\underline{x})$. Those constraints are in the form

$$\mathcal{C}(f(\underline{x}), \Phi, \nabla_{\underline{x}}\Phi, \nabla^2_{\underline{x}}\Phi, ...) = 0 \tag{2}$$

Thus, those can be enforced by minimizing a loss function depending on them, exploiting gradient descent on network parameters $\underline{\theta}$, so that $\Phi_{\underline{\theta}}$ is the solution of the problem, hence the *implicit neural representation*.

Periodic activation function Usage of periodic activations has been introduced in work by Sitzmann *et al.*[7] to improve implicit representation, basing on several works and generalization proofs. They demonstrated that this approach leads to a better implicit representation, due also to the strict relation between periodic function composition and Fourier transform (which can represent as well a

signal). For example, it has been used in Fourier neural networks[8] to provide a simpler architecture for signal representation.
A generic fully connected layer $\underline{x} \rightarrow \phi(\underline{x})$ can be expressed as:

$$\sin(\omega_0 \cdot (\boldsymbol{W}\underline{x} + \underline{b})) \tag{3}$$

where $\boldsymbol{W}$ and $\underline{b}$ are respectively the weights and the bias of the linear part. ω_0 is a factor that has been extracted to improve convergence of the problem and will be argument of discussion later on.
In addition, in [7] a successful weight initialization schema has been proposed, to adapt the network to converge faster and to deal with a sinusoidal activation function and uniform distribution of input samples, embedded in the distribution of coordinates.

Bicubic method The bicubic method is a well-known interpolation technique that does not require any kind of deep learning method. It aims to resize a generic 2D signal (upsampling or downsampling), by providing a local third-degree polynomial representation, from a 4x4 grid of original coordinates and values to correctly generate the requested points.

Enhanced Deep Residual Networks Recently, residual learning techniques exhibit excellent performance by applying the ResNet architecture to the super-resolution task. Enhanced deep super-resolution network (EDSR[4]) further improve the performance by removing the batch normalization layers from the residual blocks of ResNet architecture. This kind of techniques has clearly good results, thanks to the fact that they exploit informations acquired on several images during training.

3. Methodology

Starting from previous work about implicit representation with periodic activation functions from Sitzmann *et al.* [7], we propose a network architecture able to solve different tasks in order to replicate their results and possibly improve them. Then, by using the acquired knowledge, we aim to solve super resolution task.

Architecture The network is composed by a series of fully connected layers with sinusoidal activations, and terminated by a fully connected one. Each hidden layer has a fixed dimension of 256 neurons, in order to obtain a balance with respect to computational resources efficiency and good results, since network's parameters grows quadratically in the number of features per layer.
The network takes in input a grid with equally spaced points $\underline{x} \in \mathbb{R}^n$, where the i-th component $x_i \in [-1, 1]$ and n depends on the kind of signal we aim to learn. In particular $n \in \{1, 2\}$ respectively for audio signals and images.
We observe that for image related tasks, deepening a 2-hidden layers network does not lead to significant improvements, while dealing with audio signals requires a deeper network, due to the higher variability of this kind of signal, and the huger number of samples.

Training Considering that our final goal is to fit a specific signal, we don't need to generalize on multiple ones, but we aim to minimize the training loss, trying to overfit the signal, except in the case of super resolution where we need to generalize. We observe that in the training procedure, a scheduler is needed to adapt the learning rate to continuously minimize the loss. The best choice for all tasks seems to be the ReduceLROnPlateau[2], that often allows us to start from an higher learning rate with respect to the one of the original SIREN implementation.

Images manipulation Due to resources limitations, we are not able to work with original high resolution images from DIV2K[1]. So we resized them by a factor 2, selecting only those ones whose respective low resolution one has both dimensions even, to avoid rounding problems and consequently mismatching of scale factor 4 between HR and LR images. The resulting subset is composed by 11 images.

Loss function For our experiments we choose as a loss function the mean of the squared differences between true and predicted values. Given m the number of samples of the signal, the *Mean Squared Error* is computed as:

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(f(\underline{x}_i) - \Phi(\underline{x}_i))^2 \tag{4}$$

3.1. Audio signal fitting

For this task, we use SIREN to parameterize an audio waveform $f(t)$ at time point t by a function Φ. That is we seek the function Φ such that: $\mathcal{L} = \int_{\Omega} \|\Phi(t) - f(t)\| \mathrm{d}t$ is minimized, in which Ω is the domain of the waveform.
Since audio signals have samples in the range $[-1, 1]$, no preliminary normalization is required.
Furthermore, we decide to dynamically change the network by increasing the number of hidden layers proportionally to the number of samples and channels for each audio used.

First layer $\boldsymbol{\omega_0}$ As already studied in [7], this value can be used to better match the frequency spectrum of the signal $f(t)$. So, due to the highly periodic nature of audio signals and their high spatial frequency, we increase this parameter on the first layer to match the higher frequencies of the audio signal.
More in depth, we found that a good approximation for the first ω_0 is using the sampling rate (labelled with ω_r) of the audio signal in input, that increase the spatial frequency of the activation of the first layer.

Combining losses We also try to add another constraint to analyse its effect on the results.

[2]Find docs at `https://pytorch.org/docs/stable/optim.html#torch.optim.lr_scheduler.ReduceLROnPlateau`

While analysing the audio signal task, we change the learning phase by combining MSE (4) with the MSE of the modulus of the Fast Fourier Transform (FFT) of the true values (y) and predicted ones ($\hat{y}$):

$$MSE_{FFT} = \frac{1}{N}\sum_{i=0}^{N} \|FFT(y)_i - FFT(\hat{y})_i\|^2 \quad (5)$$

Then total MSE becomes:

$$MSE_{tot} = MSE + \frac{MSE_{FFT}}{\alpha} \quad \alpha > 0 \quad (6)$$

where $\alpha = 50$ is our chosen coefficient found to work well to weight the contribution of the MSE_{FFT}.

3.2. Image fitting

For this task, the objective is to train the SIREN in order to overfit an image, so that its reconstruction is as close as possible to the image itself. The point in performing good at it, is that this task acts as a baseline for the following ones. Indeed the images we used here are the ones in low resolution, because they are the ones we want to outperform in, to have a good starting point later in the super resolution task. For seek of generalization, we built a network able to learn a function $\Phi_{\underline{\theta}} : \mathbb{R}^2 \rightarrow \mathbb{R}^m$, being m the number of channels automatically taken from every input image.
In addition, since the distribution of the network weights is centered in 0, it is easier for the SIREN to learn a centered signal with respect to an image which has values in the interval $[0, 1]$. Thus, before training, each image is rescaled in the range $[-1, 1]$ and then the output is shifted back to the original distribution.

First layer $\boldsymbol{\omega_0}$ We observed that the initialization of the ω_0 of the first layer has a great impact on the final performance and, similarly to the audio signal task, we searched for a metric to estimate how rapidly an image varies. We supposed that a blurry image has a low variability, thus needs a lower ω_0 in the first layer, and viceversa.
Pertuz *et al.* [6] review many methods to evaluate blurriness, one of the simplest and quite effective is the one proposed by Pech-Pacheco *et al.* [5] by computing the variance of the Laplacian. This implementation gives an high value for non blurry images, and a low value for blurry ones.
Therefore, for each image we initialized the first ω_0 with the standard deviation (i.e. the square root of the variance) of the Laplacian, multiplied by a factor 250. This initialization seems to work better with respect to a fixed one with a lower value (30) proposed by [7]. Hence we defined this initialization as

$$\omega_0^* = 250 \cdot \sqrt{\text{Var}[\tilde{\Delta}_{\underline{x}} f(\underline{x})]} \quad (7)$$

3.3. Poisson equation solving

The proposed architecture is also able to reconstruct a signal f, by knowing its laplacian (Δf). This problem, known as *Poisson equation*, can be formalized as:

$$\Delta_{\underline{x}}\Phi(\underline{x}) = \Delta_{\underline{x}} f(\underline{x}) \quad (8)$$

but it can be useful to start dealing with simpler partial derivatives equations, considering the gradient ∇f:

$$\nabla_{\underline{x}}\Phi(\underline{x}) = \nabla_{\underline{x}} f(\underline{x}) \quad (9)$$

Practically this implies to setup a loss function that minimises:

$$\left\|\nabla_{\underline{x}}\Phi(\underline{x}) - \nabla_{\underline{x}} f(\underline{x})\right\| \quad \text{or} \quad \left\|\Delta_{\underline{x}}\Phi(\underline{x}) - \Delta_{\underline{x}} f(\underline{x})\right\| \quad (10)$$

We apply this method to the same subset of images we used for image fitting, to evaluate performances, by estimating gradients and laplacians of images with a numerical library (using well-known techniques, respectively *Sobel operator* and *Laplacian kernel*)[3].

Using numerical filters Gradient computed on the implicit representation ($\nabla_{\underline{x}}\Phi(\underline{x})$) has a different scale w.r.t. the *Sobel operator* (which we will denote as $\tilde{\nabla}_{\underline{x}}$) computed on image. The same effect can be encountered when dealing with *Laplacian kernel* $\tilde{\Delta}_{\underline{x}}$. As suggested by [7], analytical computation of $\nabla_{\underline{x}}\Phi(\underline{x})$ and $\Delta_{\underline{x}}\Phi(\underline{x})$ should be rescaled, respectively, by a factor 10 and 10000, to avoid this kind of mismatch.
This is due to the fact that using filters gives only an estimation of gradient and laplacian (nothing better can be done having a set of points representing an image and not an analytical representation of it). We note that this rescaling is extremely subject to the numerical library and dataset of images that are used. So, the problem may converge to an image that has only a vague form of the original one.
Since we are minimizing the distance between $\nabla_{\underline{x}}\Phi(\underline{x})$ and $\tilde{\nabla}_{\underline{x}} f(\underline{x})$, which represent slightly different quantities ($\tilde{\nabla}_{\underline{x}} \simeq \nabla_{\underline{x}}$), we try to apply *Sobel operator* also to the network's output.

$$\tilde{\nabla}_{\underline{x}}\Phi(\underline{x}) = \tilde{\nabla}_{\underline{x}} f(\underline{x}) \quad (11)$$

The same procedure can be applied to laplacian and *Laplacian kernel*. As we will report in results, better convergence of the original image can be obtained and even with less computational time and resources.
This cannot be seen as general solution, since is constrained by the library used, but we think that this can be used as a way of proceed in different problems, by applying the available numerical differential operator to get similar results.

Cauchy condition While dealing with differential equations, some condition should be enforced to reach the desired solution, so we need to introduce some Cauchy condition. We choose to enforce the reconstructed image to have

[3]Find details at `https://kornia.readthedocs.io/en/latest/filters.html`

the same mean of the original one, while dealing with first order partial derivatives equation:

$$\begin{cases} \tilde{\nabla}_{\underline{\mathrm{x}}}\Phi(\underline{\mathrm{x}}) = \tilde{\nabla}_{\underline{\mathrm{x}}} f(\underline{\mathrm{x}}) \\ \mathbb{E}[\Phi(\underline{\mathrm{x}})] = \mathbb{E}[f(\underline{\mathrm{x}})] \end{cases} \tag{12}$$

and similarly, while dealing with *Poisson equation*, another constrain is added:

$$\begin{cases} \tilde{\Delta}_{\underline{\mathrm{x}}}\Phi(\underline{\mathrm{x}}) = \tilde{\Delta}_{\underline{\mathrm{x}}} f(\underline{\mathrm{x}}) \\ \mathbb{E}[\Phi(\underline{\mathrm{x}})] = \mathbb{E}[f(\underline{\mathrm{x}})] \\ \mathbb{E}[\tilde{\nabla}_{\underline{\mathrm{x}}}\Phi(\underline{\mathrm{x}})] = \mathbb{E}[\tilde{\nabla}_{\underline{\mathrm{x}}} f(\underline{\mathrm{x}})] \end{cases} \tag{13}$$

While constraint on image's mean makes the image significantly improved (Figure 1), constraint on gradient has no particular effect on PSNR of gradient.
For example, considering the first 500 epochs, during training on gradient and laplacian we gain respectively 8 dB and 2 dB on the image. While applying the second constraint (when training on laplacian) gives a benefit of only 0.1 dB on the gradient.

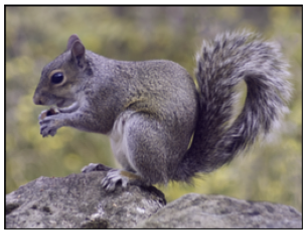
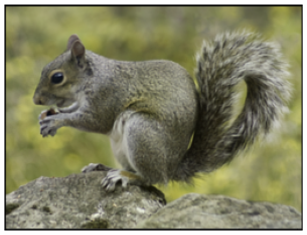

Figure 1: *Top, Left:* image reconstructed on gradient without applying the constraint. *Top, Right:* image reconstructed on gradient applying the constraint. *Bottom, Left:* original image.

3.4. Super resolution

Basing on previous tasks' analysis, now we focus our study on how the implicit representation obtained with this network can be exploited for super resolution. To achieve this, we train the network on a low resolution (LR) image, then at test time a denser grid of coordinates is given as input to the network to evaluate the representation in high resolution (HR).

Estimation of ω_{HR} Since the objective is to obtain an HR image, we aim to use its ω_0^* (7) value also during training, considering it as unavailable at learning time. In addition we observed that LR and HR images have different values of ω_0^* and there is not a direct proportion between them. Thus we tried to estimate it by looking at the unused images from the DIV2K validation dataset. Overall we found that the factor between ω_0^* of HR images and the one of LR images once unscaled to HR with the Bicubic method, is on average 0.15. Hence we use this approach to estimate ω_{HR} to use it as first layer ω_0.

Combining losses To get a more general representation, we try to use the same approach of section 5.2 combining two different losses: MSE of the image and MSE of the gradient (using *Sobel operator*) rescaled by a factor 10.

Training tricks In order to avoid overfitting on the LR image, we try to perform multiple trainings on different versions of the same image properly downsampled and/or upsampled.
Firstly, we started to train the network on reduced quality versions of the image (downsampled with *nearest* method[4] and upsampled with *bicubic*), on the LR image itself, and on some upscaled ones up to scale factor 2.
Another trick we consider is to train the network firstly on the LR image itself, and then with progressively upsampled ones by using bicubic method for small number of epochs.

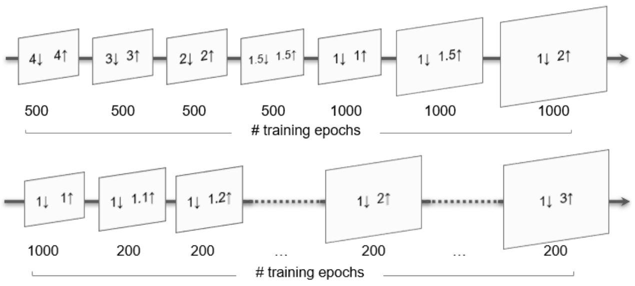

Figure 2: *Top:* trick 1 input training sequence. *Bottom:* trick 2 input training sequence. For each image the downscale (↓) and upscale (↑) factors are reported, together with the number of training epochs.

4. Ablation studies

In this section we want to focus on some parts of the network, to better understand their role in the entire architecture, by modifying them one at a time.
To test the various changes' behaviour, we will analyse the activations' distribution at each layer of a simple untrained SIREN fed with a uniform input. Then, we will run the image fitting task on a single image on 500 epochs. Results will refer to image 0803 of DIV2K[1], verifying that similar results are obtained also for other images.

4.1. First layer ω_0

By conducting this study we can see a relationship between ω_0 and the activation spectrum. In particular, Figure 3 shows that by increasing ω_0 we obtain a greater spectral

[4]Simple algorithm that keeps the closest pixel to the one to extract.

component. This relationship motivates our search of ω_0^* (7) for having the best activation spectrum and better match the frequency of the signal.

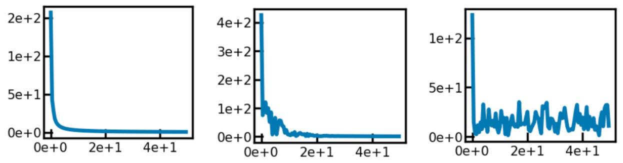

Figure 3: From left to right we have the spectrum of the output of the last layer for ω_0 equal to 1, 30 and 1000

In addition, on the image fitting task 3.2 we applied different values of ω_0 by maintaining invariant the other network configurations. The results supports our claim and referring to Figure 4 we can see good values of PSNR with ω_0 which is closer to the frequency of the signal while we lose in performance with ω_0 too small or high.

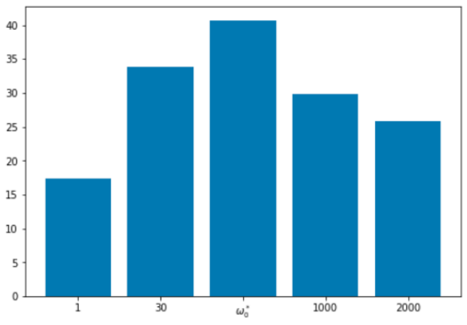

Figure 4: Results in terms of PSNR for different ω_0 for the first layer

4.2. First layer initialization

We try to change the initialization schema of the first layer from the one proposed in [7], which is a uniform distribution of weights in range $(-1/fan_{in}, 1/fan_{in})$, needed to obtain a gaussian distribution as input of the first sine non-linearity.
In our study, we consider the usage of two different techniques, commonly used in literature and evaluating their behaviour on our network.

***Xavier* initialization[2]** proposes a value for the variance of weights to use for initialization.
We use, following this approach, a uniform $\mathcal{U}(-\sqrt{6}/\sqrt{fan_{in}+fan_{out}}, -\sqrt{6}/\sqrt{fan_{in}+fan_{out}})$.
Apparently, this is not so far from the one proposed by [7], but the effect that is produced is that the output of the linear part has a data distribution that looks like noise (Figure 5).

***He* initialization[3]** is another well-known schema that, starting from results of [2], improved its performances.
We used the gaussian version, with zero mean and standard deviation $\sqrt{2/fan_{in}}$. Despite being a completely different initialization schema, this provide an initial distribution similar to the one of [7], but with a larger spectrum of the output of the network (Figure 6).

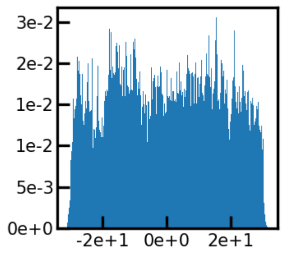

Figure 5: Distribution of data before applying first non linearity with *Xavier* initialization for the first layer

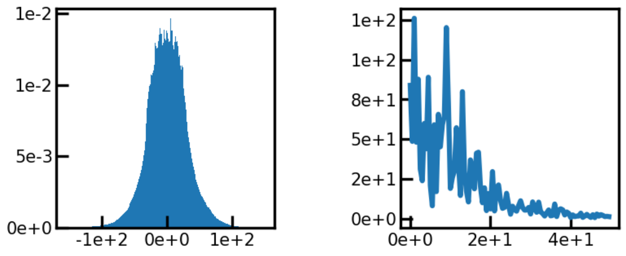

Figure 6: *Left:* distribution of data before applying first non linearity. *Right:* spectrum of the output signal.

Results on image fitting We run these experiments with a starting learning rate of $5 \cdot 10^{-4}$, to compare results in different configurations.
We also use for the first layer $\omega_0 = 30$ (as indicated in [7]) and our computation of ω_0^*.

	Sitzmann	Xavier	He
$\omega_0 = 30$	34.41	23.76	41.30
ω_0^*	41.75	33.70	36.70

Table 1: Measure of PSNR of single image with different first layer weights initializations.

Considering default initialization of ω_0, ***He*** schema works better: this can be due to the fact that providing an initial spectrum that is huger, will help in fitting the image from the beginning. When introducing ω_0^* computation, this will bring to the same effect, helping the convergence of the problem.

4.3. Hidden layer initialization

We perform, then, the same changes to initialization schema of section 4.2 but to the hidden layers instead.
In our work we used the one proposed by Sitzmann [7], a uniform distribution $\mathcal{U}(-\sqrt{\frac{c}{\omega_0 fan_{in}}}, \sqrt{\frac{c}{\omega_0 fan_{in}}}), c = 6$.
Both ***Xavier*** and ***He*** initialization schemas produce the same results in terms of activations' distributions when applied to hidden layers. While there are no alterations in the

first layer activations, starting from the first hidden sine non-linearity, the spectrum of the activation is much more larger and quite uncommon. In addition, outputs of hidden layers match a gaussian distribution but with a much higher variance. Finally, the output of the network is close to a constant, indeed its spectrum has only an high component at very low frequencies (Figure 7).

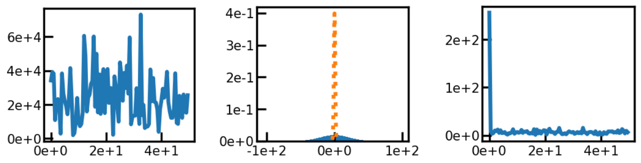

Figure 7: Reported results with *Xavier* hidden-layer initialization, the ones with *He* are very similar. *Left:* spectrum of data after applying first hidden non linearity. *Center:* output distribution of second hidden layer. *Right:* spectrum of the output signal.

We run the image fitting task, using $\omega_0 = 30$ for the first layer.

	Sitzmann	Xavier	He
$lr = 5 \cdot 10^{-4}$	34.34	9.84	8.62
$lr = 1 \cdot 10^{-4}$	26.38	20.86	18.96

Table 2: Measure of PSNR of single image with different initial learning rates for each hidden layer initialization.

We observed that with an higher learning rate, both *Xavier* and *He* initializations do not take to convergence (output images are a monochromatic spot). While with a lower one, we obtain acceptable results, but still worse than the one obtained with *Sitzmann* initialization.

4.4. Considerations

By analysing these results, we confirmed our hypothesis on ω_0^* (7) which is a significant hyperparameter for the convergence. In addition, we can validate that changing the first layer initialization does not lead to disruptive results since the distributions are changed only in the first layer. While, applying the variations to the hidden layers drastically worsen results.

5. Results

In this section we report the results of our experiments, specifying architecture and hyperparameters used.
In sections 5.2 to 5.4 we used for comparison a ReLU-based multi layer perceptron, another architecture that has been investigated in the domain of implicit representation. Its only difference with SIREN is in the use of ReLU as activation function. For this network, we kept the same weights initialization of SIREN and we set the first $\omega_0 = 30$, since we found this to be the best configuration.
In every experiment the Adam optimizer is used, combined with the ReduceLROnPlateau scheduler. The latter is in charge of decreasing the learning rate of a factor γ when the loss does not decrease for P (*patience*) epochs. We observed that we get good results with $\gamma = 0.6$ in order not to have too drastic changes, and $P = 10 \cdot$ #hidden layers.

5.1. Metrics

Results are expressed using some of the most common metrics. Specifically, the already explained *Mean Squared Error* (4) and furthermore, to express an image quality, a common metric is the *Peak Signal-to-Noise Ratio*. Given MAX_I as the maximum possible pixel value (in our case is always 1), and the MSE, it can be computed as

$$PSNR = 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \tag{14}$$

5.2. Audio signal

Here, we use three different audios: the Bach sonata, the record of a voice counting, both already used in [7], and a movie intro soundtrack[5] (chosen for its different features, that are the dual channel and a lower sampling rate). For these experiments we choose a starting learning rate equal to $1 \cdot 10^{-4}$, 500 epochs for a single run and no regularization is applied. As mentioned in the methodology part 3.1 the number of hidden layers change per each audio and in order to have a reasonable running time we decide an upper bound to 5 hidden layers. This limit is reached by Counting and Movie audios, instead for Bach we have 3 hidden layers.
In addition, for SIREN with first layer ω_0 configuration we decide to reuse the value $\omega_0 = 3000$ from the related work done by *Sitzmann* in the audio signal task. Instead, for SIREN with first layer ω_r configuration we try to use the sampling rate obtained from each audio with the following values: $\omega_r = 44100$ for Bach and Counting and $\omega_r = 22050$ for Movie.

	Bach	Counting	Movie
ReLU	$2,42 \cdot 10^{-2}$	$7,40 \cdot 10^{-3}$	$6,84 \cdot 10^{-2}$
SIREN ω_0	$5,40 \cdot 10^{-4}$	$1,06 \cdot 10^{-3}$	$1,41 \cdot 10^{-2}$
SIREN ω_r	$3,80 \cdot 10^{-4}$	$7,00 \cdot 10^{-4}$	$8,36 \cdot 10^{-3}$
SIREN $\boldsymbol{\omega_r}$ & $\boldsymbol{MSE_{tot}}$	$2,40 \cdot 10^{-4}$	$6,40 \cdot 10^{-4}$	$8,62 \cdot 10^{-3}$

Table 3: Measure of MSE in avarage over 5 independent runs per audio. The best configuration is the last one with a further improvement by using MSE_{tot}(6) loss

[5]Available at `https://soundbible.com/1676-Movie-Start-Music.html`

5.3. Image fitting

Here we report the results of the image fitting task. We trained each image for 5000 epochs, with a 2-hidden layers network. The initial learning rate is set to $5 \cdot 10^{-4}$ for SIREN, and $1 \cdot 10^{-4}$ when using ReLU, we observed that we get better results with these values. No regularization is needed, since our goal is precisely to overfit an image.

	Mean	Std
SIREN	41.62	3.55
ReLU	14.26	1.99
SIREN with ω_0^*	57.18	10.10

Table 4: Measure of PSNR (mean,std) on selection of LR DIV2K for image fitting

As discussed before, the initialization of ω_0 of the first layer has a huge impact on results (Figure 8). Indeed, despite the high variability, we obtain very good results by using the ω_0^* initialization. Depending on the image we also reach peaks of 80 dB of PSNR.

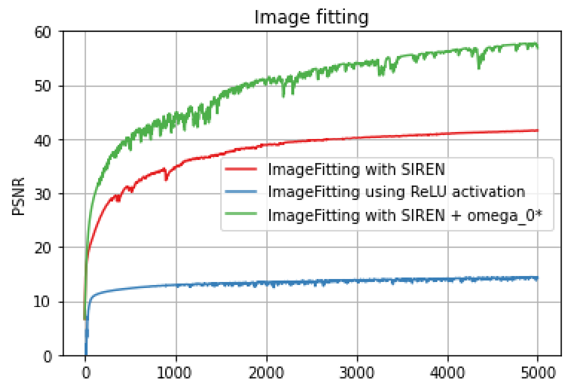

Figure 8: Mean of PSNR of the selection of LR DIV2K images computed along the 5000 epochs of the training. As can be seen, the ω_0^* initialization leads to better results with a faster convergence.

5.4. Poisson equation solving

We run our 2-hidden layers architecture (and the corresponding ReLU version) for 5000 epochs on all the dataset, considering the usage of ω_0^* (only when using SIREN, while with ReLU 30 is used as the default value) considered at previous point. We use a starting learning rate of $1 \cdot 10^{-4}$ (which has been augmented to $5 \cdot 10^{-4}$ while dealing with SIREN numerical), with no regularization.
We will report results for experiments matching the gradient of the image, both using numerical filters and analytical methods (used in [7]) on network output.
When dealing with analytical results, we find that, due to the library we are using, these problems will converge to visually acceptable solutions when multiplying gradient and laplacian, respectively, by factors 2.5 and 0.05 (second one does not bring to perfect results, but it is the one that leads to the best-looking image).

5.4.1 Training on gradient

	Image		Gradient		Laplacian	
	Mean	Std	Mean	Std	Mean	Std
SIREN, analytical	18.88	4.46	27.18	3.57	2.33*	2.19*
SIREN, numerical	33.88	5.21	45.33	3.61	18.00	3.15
ReLU, numerical	12.90	3.34	1.26	2.21	2.32	2.15

Table 5: Measure of PSNR (mean, std) on selection of LR DIV2K trained on gradient. *these data can be biased from the coefficient 0.05 we found to compare laplacian kernel and analytical.

Very significant time improvements are obtained (measures are reported as average training time per image):

- using analytical: 9 min 33 s.
- using numerical (both for ReLU and Siren): 1 min 52 s.

It can be noticed that results obtained with our technique are comparable with the one of [7], but they are obtained with half of the epochs and in an extremely reduced amount of time.

5.4.2 Training on laplacian

Considering the results obtained in training on gradient, we will consider the solution of using numerical filters, avoiding to run a training that can last hours for a single image on laplacian (time performances during this training drastically worsen with analytical method).

	Image		Gradient		Laplacian	
	Mean	Std	Mean	Std	Mean	Std
SIREN	10.34	3.51	14.39	3.88	36.31	4.55
ReLU	−13.19	4.77	−2.39	2.18	2.13	2.17

Table 6: Measure of PSNR (mean, std) on selection of LR DIV2K trained on laplacian

A ReLU-based architecture is not able to solve this kind of task, while usage of periodic activation function get nice results in limited amount of epochs.

Results obtained with SIREN architecture are a bit lower than the ones of [7], but our results are obtained with a simpler network, with half of the epochs and are computed on RGB images.

5.5. Super resolution

We compare our results with some state-of-arts methods already mentioned before in section 2. For Bicubic we used the implementation provided by `torch` library, while for the EDSR method we took the implementation by Krasser *et al.*[6] pretrained on DIV2K.
All SIREN networks have a fixed configuration with 2-hidden layers, ω_{HR} for the first layer, starting learning rate of $5 \cdot 10^{-4}$, regularization factor of $5 \cdot 10^{-6}$ and a combined loss (section 3.4). They all have been trained for 5000 epochs, eventually partitioned on different versions of the image itself.
All experiments has been done on the selection of images from LR DIV2K as before, with the goal of obtaining an HR with a scale factor of 4 w.r.t LR .

	Mean	Std
Bicubic	20.36	2.68
EDSR	26.58	3.50
Basic SIREN	19.42	2.67
SIREN trick 1	20.23	2.55
SIREN trick 2	20.32	2.55

Table 7: Measure of PSNR (mean,std) of reconstructed SR images starting from selection of LR DIV2K

We can observe that results obtained through SIREN-based networks are not able to reach Bicubic interpolation method, even though training tricks partially exploit it while upsampling images.
Our hypothesis is that this is due to an intrinsic limitation of SIREN, since it is based on an implicit representation of a signal, that is practically an interpolation of the signal itself. This also explains similar results obtained with Bicubic, which has a common mathematical foundation with SIREN.
To get better results a more complex network is needed, to acquire knowledge from other images, following the EDSR and similar methods' approach based on CNN.

6. Future works

Further studies could be done on our work. Firstly, a more detailed analysis on ω_0^*, testing other indexes, providing a formal description of the problem and trying to retrieve a generalized approach valid for any kind of signal.
Then, the role of the number of hidden layers and hidden features could be better investigated, since from our tests we did not observe any successful result in varying them.
In addition, some work can be done trying to find a more successful training trick for super resolution, even though we already highlighted the limitations of this network in this task.
Possibly, a research can be done to formalize a way to retrieve the factors between analytical gradient/laplacian and respective numerical filters to solve Poisson equation.

7. Conclusions

Basing on the obtained results, we can state that this architecture has very good performances on fitting signals and solving differential equations. With our work we showed some improvements that can be done to reach good results, underlying also what can be furtherly studied.
On the other side, we described that SIREN is not able to outperform in problems where a generalization is needed, since it suffers the same problems of interpolation methods.

References

[1] E. Agustsson and R. Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[2] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research - Proceedings Track*, 9:249–256, 01 2010.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015.

[4] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee. Enhanced deep residual networks for single image super-resolution. *CoRR*, abs/1707.02921, 2017.

[5] J. L. Pech-Pacheco, G. Cristobal, J. Chamorro-Martinez, and J. Fernandez-Valdivia. Diatom autofocusing in brightfield microscopy: a comparative study. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 3, pages 314–317 vol.3, 2000.

[6] S. Pertuz, D. Puig, and M. Á. García. Analysis of focus measure operators for shape-from-focus. *Pattern Recognit.*, 46:1415–1432, 2013.

[7] V. Sitzmann, J. N. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein. Implicit neural representations with periodic activation functions. In *Proc. NeurIPS*, 2020.

[8] A. Zhumekenov, M. Uteuliyeva, O. Kabdolov, R. Takhanov, Z. Assylbekov, and A. J. Castro. Fourier neural networks: A comparative study. *CoRR*, abs/1902.03011, 2019.

[6]Implementation can be found at `https://github.com/krasserm/super-resolution`