Implicit Neural Representation with periodic activation functions

Matteo Abbamonte* s277483 Paolo Gastaldi* s277393 Stefano Gennero* s270227 Alkis Koudounas* s278266

Abstract

We present a 2 hidden layers SIREN implementation that achieves good performance in representing complex natural signals and their derivatives. We exploit the strength of this model, proving that it is suited for fitting images and reconstructing them starting from their first or second-order derivatives. Moreover, we show the fundamental role played by the initialization scheme in the effective training of the model. Lastly, we use our model to address the SISR (Single Image Super-Resolution) task, also trying to make some changes to the standard implementation and comparing the obtained outcomes with state-of-the-art results.

1. Introduction

The concept of Neural Network is gaining popularity nowadays, and it is changing the way in which people and organizations interact with systems, solve problems, and make better decisions and predictions. Neural networks can learn and model the relationships between inputs and outputs that are nonlinear and complex, but also reveal hidden relationships, patterns and predictions.
Different models have also been thought for the representation and classification of signals. In this sense, implicit neural representation is a novel way to parameterize signals of all kinds. Conventional signal representations are usually discretized by the used models. In contrast, implicit neural representations parameterize a signal as a continuous function that maps the domain of the signal (i.e., a coordinate) to whatever is at that coordinate (for an image, an RGB value). Implicit models have a lot more expressive power than standard networks, and most importantly they are not coupled to spatial resolution anymore (i.e., the way in which an image is coupled to the number of pixels), because they are in the form of continuous functions. Thus, the memory required to parameterize the signal is independent of the spatial resolution, and only scales with the complexity of the underlying signal. Implicit representations have *infinite resolution*, which means that they can be sampled at arbitrary spatial resolutions. This is useful for a number of applications, such as super-resolution, or in parameterizing signals in 3D and higher dimensions.

*Politecnico di Torino, sxxxxxx@studenti.polito.it

The authors of [8] propose to leverage periodic activation functions for implicit neural representations and demonstrate that these networks, dubbed SIRENs, are suited for representing complex natural signals and their derivatives. The usage of the sine activation function enables the implicit representation to be infinitely derivable and therefore both signals and their derivatives can be well represented.
In this paper we present an implementation of a 2 hidden layers SIREN network and we show several results obtained while performing well-known tasks, such as Image Fitting and Image Reconstruction (i.e., solving Poisson equations on gradients and laplacian). Finally we carry out various ablation studies regarding the configurable parameters of the network, and we use our model for a SISR (Single Image Super-Resolution) task, comparing the outcomes with state-of-the-art results.

1.1. Dataset

All the experiments reported have been performed on three different datasets, namely Skimage [9], BSDS500 [5] and Div2K [1]. BSDS500 dataset consists of 500 natural images, ground-truth human annotations and benchmarking code, and the data is explicitly separated into disjoint train, validation and test subsets. Div2K contains 1,000 2K resolution images divided into 800 images for training, 100 images for validation and 100 images for testing.
Specifically, Image Fitting relies on the *Camera* image of the Skimage dataset, that is also used for the SISR task. Image Reconstruction is fulfilled on the *Starfish* image of the BSDS500 dataset, center-cropped to 321×321 and resized to 256×256. Finally, SISR task is carried out using both the *Camera* image of the Skimage dataset and several images from the Div2K dataset.

1.2. Metrics and techniques

We refer to paper [10] for the metrics' usage. Training procedures are based on the MSE function (1), that is a pixel-wise metric which tends to achieve high PSNR results.

$$MSE = \frac{\sum_{i=0}^{D}(x_i - \hat{x}_i)^2}{D}, \; D = number\ of\ pixels \quad (1)$$

Regarding the comparison, our main metric is the Peak Signal to Noise Ratio, PSNR (2), which compares each predicted pixel with the corresponding one on the ground-truth

image. As this metric does not reflect in a good way what the human eye perceives, we also use the SSIM (Structural Similarity Index Measure) metric (3).

$$PSNR = 20 \log_{10} \frac{max\ pixel\ value}{\sqrt{MSE}} \tag{2}$$

$$\begin{aligned} &SSIM = \frac{2\mu_x\mu_{\hat{x}} + k_1}{\mu_x^2 + \mu_{\hat{x}}^2 + k_1} \cdot \frac{\sigma_{x\hat{x}} + k_2}{\sigma_x^2 + \sigma_{\hat{x}}^2 + k_2}, \\ &where: \\ &\mu = mean, \sigma = covariance, \sigma^2 = variance \\ &k_1, k_2 = constant\ relaxation\ terms \end{aligned} \tag{3}$$

2. Related Works

A promising recent direction in computer vision replaces traditional discrete representations of signals with continuous functions parameterized by deep neural networks. These architectures, called implicit neural representations, take as input the spatio-temporal coordinates and are trained to output a representation of the signal at each input location. The purpose of this section is to acknowledge the benefits offered by the SIREN [8] innovative approach (subsection 2.1), as well as the novel NeRF [6] model used to represent a static scene as a continuous 5D function that outputs the radiance emitted in each direction (θ, ϕ) at each point (x, y, z) in space (subsection 2.2). Furthermore, we exploit the SISR (Single Image Super-Resolution) task [10], that aims to obtain a high-resolution output from one of its low-resolution versions (subsection 2.3), analyzing also a state-of-the-art model, SRGAN [4], that is the first framework capable of inferring photo-realistic natural images for $\times 4$ upscaling factors (subsection 2.4).

2.1. SIREN

Being interested in implicit neural representation problems, the attention is focused on a class of functions Φ that satisfies equations of the form:

$$F(x, \Phi, \nabla_x \Phi, \nabla_x^2 \Phi, ...) = 0, \quad \Phi : x \mapsto \Phi(x) \tag{4}$$

This problem formulation needs as input the spatial or spatio-temporal coordinates $x \in \mathbb{R}^m$ and, optionally, the derivatives of Φ with respect to these coordinates.
The authors of [8] propose a neural network architecture that uses the sine as a periodic activation function:

$$\begin{aligned} \Phi(x) = W_n(\phi_{n-1} \circ \phi_{n-2} \circ ... \circ \phi_0)(x) + b_n, \\ x_i \mapsto \Phi(x_i) = sin(W_i x_i + b_i) \end{aligned} \tag{5}$$

where $\phi_i : \mathbb{R}^{M_i} \mapsto \mathbb{R}^{N_i}$ is the i^{th} layer of the network, consisting of the transform defined by the weight matrix $W_i \in \mathbb{R}^{N_i \times M_i}$ and the biases $b_i \in \mathbb{R}^{N_i}$, applied on the input $x_i \in \mathbb{R}^{M_i}$, followed by the sine non-linearity applied to each component of the resulting vector. The strength of this idea regards the fact that any derivative of a SIREN is a SIREN itself, since the derivative of the sine is a phase-shifted sine (i.e., cosine). Providing an initialization scheme that preserves the distribution of activations throughout the whole network, they present a model that does not suffer from either vanishing or exploding gradients at initialization. In particular, they propose to draw weights according to a uniform distribution $W \sim \mathcal{U}(-\sqrt{6/fan_{in}}, \sqrt{6/fan_{in}})$, where fan_{in} is the number of input units in the weight tensor. In this way, the input of each sine activation is Gauss-Normal distributed, whereas the output of each sine activation is approximately arcsine-distributed with a standard deviation of 0.5.
Furthermore, in the first layer Sitzmann et al. introduce a factor ω_0 that increases the spatial frequency of the layer itself to better match the frequency spectrum of the signal. By that means they also find the cited ω_0 interesting for accelerating the training of the SIREN, by factorizing the weight matrix W as

$$W = \hat{W} * \omega_0, \ \hat{W} \sim \mathcal{U}\left(-\sqrt{\frac{6}{\omega_0^2 fan_{in}}}, \sqrt{\frac{6}{\omega_0^2 fan_{in}}}\right) \tag{6}$$

This helps in keeping the activations' distribution constant, but it also boosts gradients to the weight matrix $\hat{W}$ by the factor ω_0. The authors demonstrate that SIREN networks behave perfectly for the representation of images, wave fields, video, sound, and their derivatives, as well as in solving challenging boundary value problems, such as the Poisson, the Helmholtz and wave equations.

2.2. NeRF

As described in paper [6], the Neural Radiance Field (NeRF) technique is used for a better fitting over signals with high frequencies. For 2D image inputs, this is used with complex scenes for a faster convergence to a high-resolution representation. There are two main approaches: positional encoding and hierarchical sampling. While the latter is mainly used for 2D image sampling of 3D objects, the former can be applied more easily. The positional encoding is a dimensional expansion of the input before it is sent to the original network. The encoding function applied (7) performs a mapping into a higher dimensional space so that the MLP can better approximate the function.

$$\begin{aligned} \gamma(p) = (&\sin(2^0 \pi p), \cos(2^0 \pi p), \ldots, \\ &\sin(2^{L-1} \pi p), \cos(2^{L-1} \pi p)) \\ L = &\begin{cases} 4, & with\ 2D\ input \\ 10, & with\ 3D\ input \end{cases} \end{aligned} \tag{7}$$

According to the cited paper, a network using these techniques can improve the fitting, in particular over real im-

ages. A smoother light perception and a better representation of high-frequency image sections can be noticed in the obtained results.

2.3. SISR

The Single Image Super-Resolution (SISR) problem is one of the most challenging tasks and applications of machine learning. Super-resolution (SR) refers to the task of reconstructing high-resolution images from one or more low-resolution observations of the same scene [10]. The SR can be classified into single image super-resolution (SISR) and multi-image super-resolution (MISR), based on the number of input LR images. In this section we focus on the first one, the SISR task. In the typical SISR framework, the LR image y is modeled as follows:

$$y = (x \otimes k) \downarrow_s + n, \tag{8}$$

where $x \otimes k$ is the convolution between the blurry kernel k and the unknown HR image x, $\downarrow_s$ is the downsampling operator with scale factor s, and n is the independent noise term. Solving (8) is an ill-posed problem, because one LR input may correspond to many possible HR outputs.
SISR techniques are grouped in three main categories: interpolation-based methods, reconstruction-based methods and learning-based methods.
One of the most used interpolation-based methods is the bicubic interpolation [3]. This technique is popularly used as first comparison with the obtained results because it's fast and straightforward. Reconstruction-based SR methods often embrace prior knowledge to restrict the possible solution space with an advantage of generating flexible details, but in doing so their performance degrades rapidly when the scale factor increases. Finally, learning-based methods are usually based on convolutional layers as they work well with images. From that, the initial part of many deep learning networks is made by CNN layers. Also called example-based methods, these are becoming more and more important due to the fast computation and performance that they are able to achieve.
For more details we invite the reader to look through [10].

2.4. SRGAN

The SRGAN [4] is one of the SISR learning-based methods that reaches the highest performance because of its capability of representing high-frequency contents. It consists of a GAN model that is composed by two networks: a generative network and a discriminative one. Both networks are composed, for the initial part, by a ResNet with skip-connections. The generative network uses Parametric ReLU as activation function, while the discriminative network uses Leaky ReLU.
The loss function is a VGG perceptual loss (9), that is the weighted sum of the VGG loss and the adversarial loss. The latter is added with the goal of reproducing more natural images, while the former substitutes the MSE loss (1), since this encounters difficulties in representing high-frequency content and it results in producing lower quality outcomes. Instead of relying on pixel-wise loss, VGG loss defines a euclidean distance between the feature representations of a reconstructed image $G_{\theta_G}(I^{LR})$ and the reference image I^{HR}.

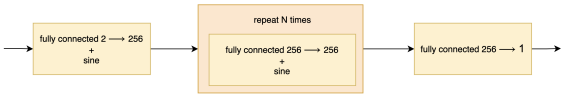

Figure 1. The SIREN architecture, with 256 features. Our initial version has 2 hidden layers (N=2), while the original net [8] has N=3.

$$l^{SR}_{VGG/i,j} = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} (\phi_{i,j}(I^{HR})_{x,y} + - \phi_{i,j}(G_{\theta_G}(I^{LR}))_{x,y})^2 \tag{9}$$

$\phi_{i,j}$ is the feature map obtained by the j^{th} convolution (after activation) before the i^{th} maxpooling layer within the VGG19 network [7], which is considered given. $W_{i,j}$ and $H_{i,j}$ are the dimensions of each feature map.

3. Experiments

3.1. Implementation of a SIREN model

We decide to implement a simple SIREN-based network [8] with 2 hidden layers of 256 features each, and an initial ω_0 equal to 30 for Image Fitting and Image Reconstruction, and then change it a bit for the following studies. Fig. 1 describes the overall architecture of the model. For more implementation details, please rely on our code [1].
We train it using ADAM optimizer with a learning rate of $1{\times}10^{-4}$, and the number of iterations depends on the performed task. The network is trained using NVIDIA Tesla T4 GPU with 16 GB of memory and 2560 cores, and it is fed with a grid that has the same resolution as the target image.

3.1.1 Image Fitting

The most simple representation task consists in fitting an implicit neural representation $\Phi : \mathbb{R}^2 \mapsto \mathbb{R}^3, x \mapsto \Phi(x)$ to an image [8]. For addressing this task we use the MSE loss (1) between the predicted image and the provided ground-truth image. The results reported in Fig. 2 show the strength of our SIREN model, that is able to achieve better outcomes with respect to a classic ReLU-based one.

3.1.2 Poisson Image Reconstruction

In this section we want to replicate the same experiments shown in [8], regarding the reconstruction of images from

[1] https://github.com/paologastaldi-polito/siren

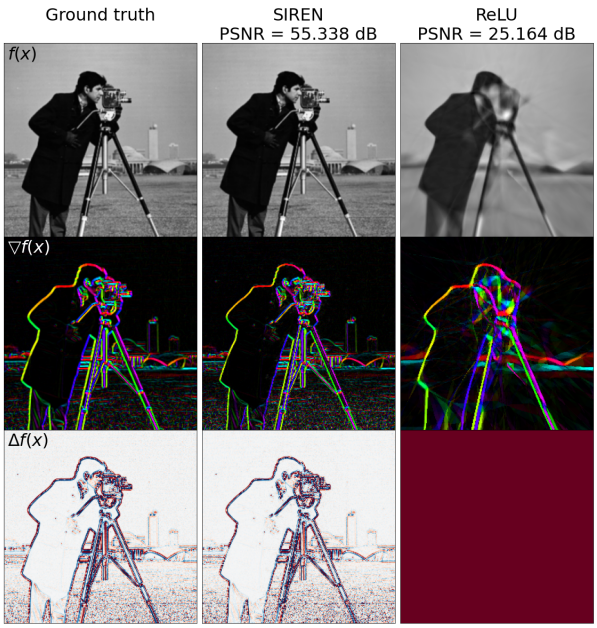

Figure 2. **Image Fitting:** Comparison of our 2 hidden layers SIREN and ReLU networks fitting the *Camera* image in a 15,000 iterations training process. The representation is only supervised on the image, but we also visualize the gradients $\nabla f(x)$ and Laplacians $\Delta f(x)$ of the image in rows 2 and 3, respectively.

their derivatives, with our 2 hidden layers SIREN model. The robustness of this model lies in the fact that it is not only able to accurately represent a function and its derivatives, but it can also be supervised merely by its derivatives.
Poisson equation represents a simple example that strengthens what we stated above, depicting an unknown ground truth signal f that is estimated from discrete samples of either its gradients ∇f or Laplacian Δf as:

$$\mathcal{L}_{gradient} = \int_{\Omega} \|\nabla_x \Phi(x) - \nabla_x f(x)\| \, dx \qquad (10)$$

$$\mathcal{L}_{laplacian} = \int_{\Omega} \|\Delta \Phi(x) - \Delta f(x)\| \, dx \qquad (11)$$

Gradient supervision In order to accomplish the task, we supervise the implicit representation with ground truth gradients via $\mathcal{L}_{gradient}$ (10). The gradient is obtained by applying a Sobel Filter on the original image, and by scaling it by a factor of 10, with the aim of having a more perceptible outcome (Fig. 3), as in [8].

Laplace supervision We supervise here the ground truth laplacians via $\mathcal{L}_{laplacian}$ (11). The laplacian is obtained by applying a N-D Laplace Filter on the original image and by scaling it by a factor of 10,000 (following the process described by Sitzmann et al.).

3.2. Comparison with ReLU based MLP

In this section we analyze the differences between our network and a simple MLP based on ReLU activation function,

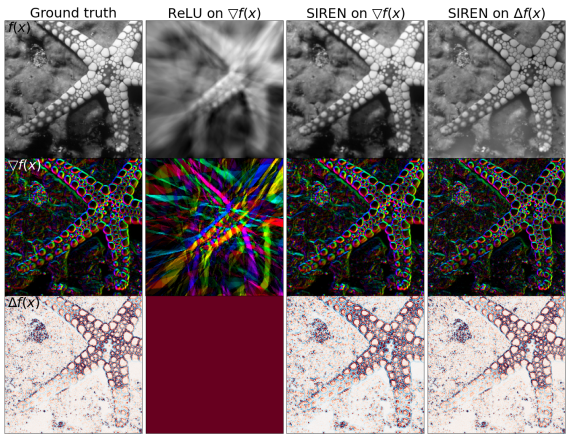

Figure 3. **Poisson Image Reconstruction:** *Starfish* image reconstructed by fitting our 2 hidden layers SIREN, supervised either by its gradients or Laplacians, and by fitting a ReLU network (supervised only by its gradients). The training process consists of 15,000 iterations.

in the fields of Image Fitting and Image Reconstruction.
In order to make the comparison fair, for what concerns the ReLU-based MLP, we keep the same number of layers and features presented in the subsection 3.1 and we change the weight initialization to a Kaiming normal one, because the one used for SIREN was too tailored on its initialization parameters. For what concerns the training phase, we use ADAM optimizer with a learning rate of 1×10^{-4}, 15,000 iterations and the MSE loss.
Fig. 2 and 3 show how our SIREN model achieves better visually perceptible results with respect to the ReLU one, not only for the Image Fitting task but especially for the Poisson Image Reconstruction, being the ReLU completely unable to represent second-order derivatives. More details can be found in the supplementary section S1.1.

3.3. Ablation studies

In the coming section we conduct three ablation studies with the purpose of understanding the important features of the network and their contribution to the overall system.
In subsections 3.3.1 and 3.3.2 we examine how to change the initialization of the weights, respectively, of the first and the hidden layers of the network. Finally, in subsection 3.3.3 we inspect how the ω_0 initialization impacts the frequency of the sine activation.

3.3.1 First layer weights initialization

As the authors of [2] suggest, initialization schemes are pivotal in the training procedure of deep neural networks. This is why [8] proposes an initialization scheme that preserves the distribution of activations through the several layers. Our ablation study in this sense aims at recognizing both how powerful is the initialization of the first layer weights in getting good results, and therefore a PSNR comparable

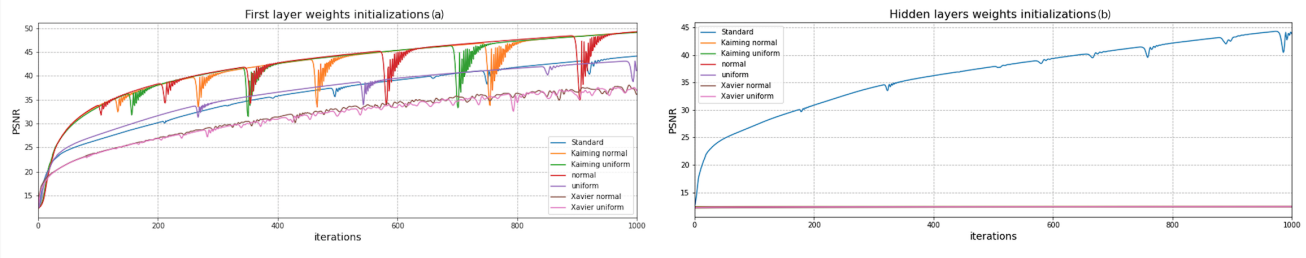

Figure 4. Comparison of the PSNR values trends on 1,000 iterations, obtained by using different initialization schemes for the first (a) and the hidden (b) layers' weights. We use 3 hidden layers for the SIREN model and it is trained on the *Camera* image.

with the values previously calculated, but also how important is this initialization in keeping the activations' distributions unchanged as the depth of the network grows.
For this study we refer to an architecture based on 3 hidden layers, but similar results can be observed also with 2 hidden layers. Fig. 4a compares some PSNR values' trends, obtained by using several initialization schemes. We can easily visualize how Kaiming initialization distributions enable the model to reach, along with normal distribution, higher peak values with respect to the scheme proposed by Sitzmann et al., whereas Xavier ones lead to poor performance. Considering also the trend of the curve throughout the 1,000 iterations process, we can state that by using the original initialization distribution for the first layer's weights it remains stable, similarly to the uniform distribution, whereas Kaiming ones perform a more unsteady behaviour. Xavier normal and uniform distribution act stable during the first steps, becoming then fairly unstable.
By also plotting the activation distributions and their relative spectra (Fig. S3 of the supplements), the inner consequences of the initialization scheme can be inspected. Specifically, the Kaiming uniform distribution exhibits an activation distribution that, after the dot product between the weights and the input values, resembles a Gauss-Normal distribution and, after the application of the non-linearity, matches perfectly the distribution showed by Sitzmann et al. The activation spectrum conveyed after the non-linearity seems more unstable with respect to the SIREN standard one, whereas for the following layers it acts closely. The Xavier uniform distribution instead manifests a high-noise distribution right after the dot product, although from the non-linearity of the first layer on its behaviour looks like the original one. The activation spectrum shows that frequency components similarly remain comparable.

3.3.2 Hidden layers weights initialization

In order to confirm the previous statement regarding the unchanging behavior of the activation distributions also with a growing-depth network, some variations to the input distributions of the hidden layers are performed, analyzing them both from the PSNR trend perspective and from the activation distribution one, together with the spectrum.

As the authors of [8] state, when each component of the weights matrix is uniformly distributed, then the output of each set of dot products converges to a normal distribution. After having applied the characteristic sine non-linearity we can observe an arcsine distribution as a result. This specific swap between normal and arcsine distribution can be considered recurring among all internal layers of the network.
By keeping the same 3 hidden layers architecture as the previous section and a standard initialization for the other parameters (firstly ω_0), if we replace the distribution of the hidden layers weights, especially with non uniform ones, a very bad behavior is revealed. This is due to the presence of ω_0 also in the internal non-linearities which, as it is described by Sitzmann et al., has been found to increase the training speed. Furthermore, the official SIREN ω_0 initialization provides a factorization of the weights matrix related to the hidden layers, as reported in 3.3.3, that keeps a good behavior also for the activation spectrum.
Our results (Fig. 4b) show an almost *flat* trend of the PSNR values related to all the tested distributions applied to the hidden layers weights, except from the default one. The same observations can be done regarding the activation distributions and spectra, in fact (Fig. S4 of supplements) the Kaiming and the Xavier uniform distributions clearly induce an unstable behaviour, differently from the stable and mathematically predictable SIREN's one. All these considerations seem to carry out the idea that a wrong initialization of the activation distribution of the hidden layers weights can cause the network to suffer from exploding gradients.
In this case, if we decrease the number of hidden layers to 2, we can observe that the network is still able to perform image fitting tasks but with poor results in terms of perceived noise (see supplements subsection S1.2 for more details).

3.3.3 Initialization of ω_0

With the aim of better addressing a wider analysis on the various parameters that can be configured on the SIREN model, in this section we try to understand the influence of ω_0 by applying some variations on it.
Based on the authors' description of [8], this parameter is crucial for increasing the spatial frequency of the first layer, in order to better match the frequency spectrum of the sig-

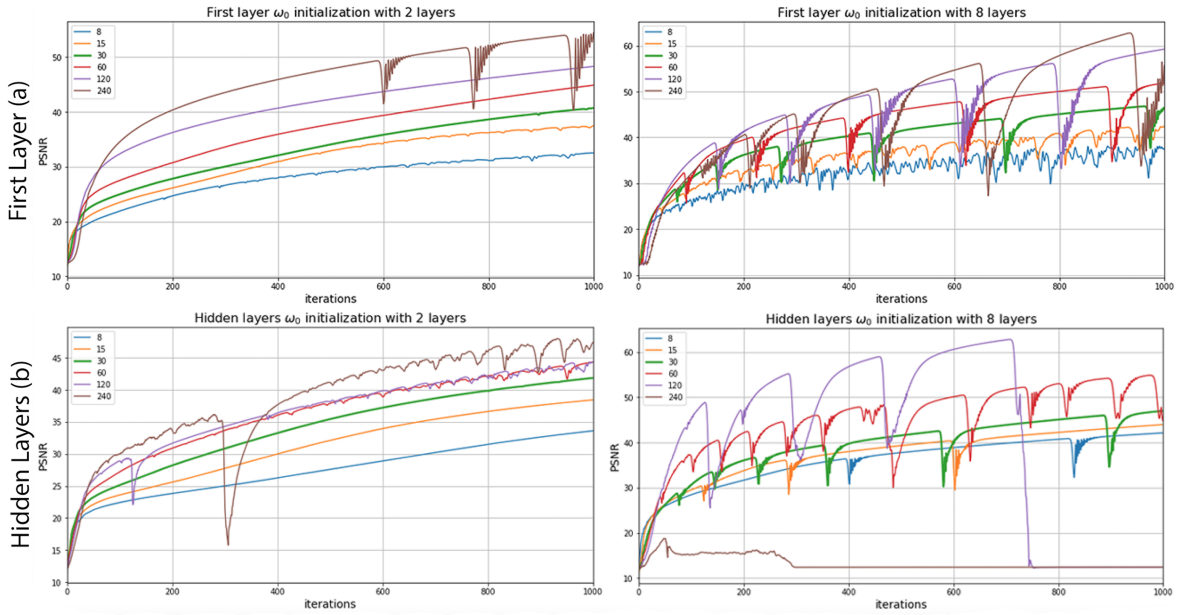

Figure 5. Comparison of the PSNR values trends on 1,000 iterations, obtained by initializing ω_0 of the first layer (a) and hidden layers (b) with values from different order of magnitude. Training on the *Camera* image.

nal. They also show how ω_0 can be used to accelerate the training of the model by parameterizing the weight matrix as $W = \hat{W} \times \omega_0$, as reported in subsection 2.1.

Given that, we try to change the value of ω_0 within a range that goes through two orders of magnitude, with the aim of observing which are the implications over the stability of the activation distributions and the PSNR values trend, both for the first layer and the hidden ones.

Regarding the first layer, Fig. 5a shows that ω_0 initialization behave differently for a 2 hidden layers SIREN and a 8 hidden layers one. Indeed, while the former exhibits a stable trend of the PSNR values, showing that for $\omega_0 = 240$ we obtain by far the best performance (even if it is unstable after 600 iterations), the latter conveys a very failing course, indicating that, as Sitzmann et al. suggest, $\omega_0 = 30$ gets more stable results as the depth of the networks grows.

With reference to the hidden layers, a statement similar to the one above holds: Fig. 5b shows how stable is the trend induced by the recommended value of ω_0 with respect to the other ones. Moreover, a higher irregularity in the drops of the PSNR values can be observed.

3.4. SISR task

Lastly we try to use our network to address a SISR task, that is the process of getting a high-resolution image starting from its low-resolution version. Specifically, we present our model and some variants in subsection 3.4.1. Subsection 3.4.2 shows the comparison between our outcomes and state-of-the-art results. For further details please take a glance at section S1.3 of the supplementary material.

Algorithm 1: Our SIREN network for a SISR task

Result: HR image from a LR one

1. Import an image at full (high) resolution as HR ground truth *HR-gt*
2. Downsample *HR-gt* and use it as LR ground-truth *LR-gt*
3. Generate a grid *LR-grid* of values in [-1, 1] with the same sidelength of *LR-gt* as the input for the model
4. Perform fitting on *LR-gt* by using *LR-grid* as input and MSE as loss function between *LR-gt* and the model output *LR-img*
5. Generate a grid *HR-grid* of values in [-1, 1] with the same sidelength as *HR-gt*
6. Use *HR-grid* as the input of the model to obtain a super-resolution image *HR-img* as output
7. Compare *HR-gt* and *HR-img*

3.4.1 Our SIREN SISR

Algorithm 1 shows the procedure for using our SIREN model applied to a super resolution task. We train the network for fitting a low-resolution image with a corresponding low-resolution coordinates' grid as input, then we feed the network with a higher resolution grid in order to exploit the implicit representation capability of the model in the super resolution field.

With MSE loss A simple SIREN model is used as a base for each following study: the standard version of the model

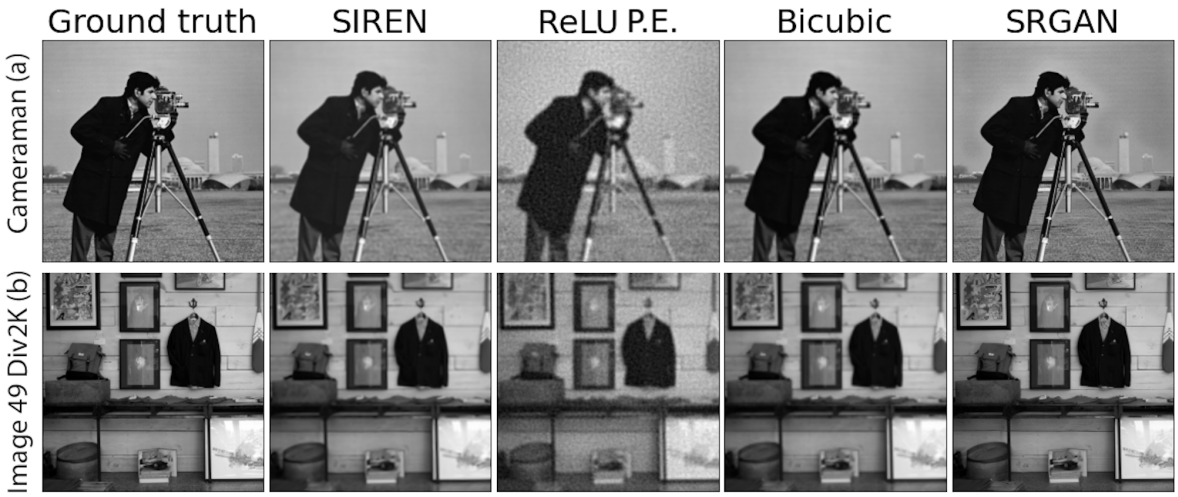

Figure 6. Comparison of *Camera* (a) from *Skimage* dataset and *Image 49* (b) from *Div2K* dataset from 128×128 to 512×512 resolution. SIREN and ReLU have 3 hidden layers with 512 hidden features each. Training process of 500 steps.

involves a MSE loss computation, applied pixel by pixel, between the predicted output and the ground truth instance of the target image. This simple version of the loss encounters some problems when facing with higher scaling factors or higher frequency detailed images. This is why, in the following, we try out several changes to modify this behaviour.

With VGG loss Starting from a pre-implemented code[2], a VGG loss (9) based on a VGG19 is implemented. Applying this loss to a SIREN network results in more time needed for fitting the image, but also in slightly increased performances on the initial steps with respect to the standard SIREN. Also gradients and laplacian get a small benefit. The network is fed with the coordinates of an image characterized by a user-defined resolution and then, in order to be compatible with the modified loss, the model output is rescaled within the loss computation.

With P.E. as in NeRF Based on [6], we try to adapt the positional encoding to our SIREN network in order to understand whether it could be useful in improving the capability of the model in retaining finer details. In fact mapping the inputs to a higher dimensional space before passing them to the real network should enable better fitting for high frequency variations in the submitted data. However, after a slightly perceptible advantage in the PSNR values in the early stages of the training process, the introduction of this technique leads to results that are comparable with the standard SIREN architecture.

3.4.2 Models Comparison

Table 1 and Fig. 6 show the comparison between PSNR and SSIM values obtained using our SIREN and some further methodologies, such as bicubic interpolation and other architectures, including state-of-the-art SRGAN network.

Bicubic As described in section 2.3, the bicubic interpolation is used as comparison for many experiments. This solution is widely adopted and it enables to reach good results, some of them comparable with most of our attempts in terms of PSNR values and visual appearance. Besides the values reported in Table 1, a visual idea of the result is available in the supplementary section.

ReLU with P.E. as in NeRF As described in [6], we apply the positional encoding technique in order to better approximate higher frequency functions in MLP-based implementations. Firstly a usual ReLU-based MLP configuration with positional encoding is trained for low-resolution image fitting, then it is fed with a denser grid. The results are almost comparable with the standard version of the SIREN, taking advantages in terms of convergence speed only in the initial steps of the training process.

SRGAN A pre-trained model script[3] of SRGAN [4] is used by exploiting the provided test mode for feeding a SRGAN with a high resolution image to be used both as ground truth for the metrics and as base for the downsampled version that is submitted to the Generator for the super resolution task. As shown in Table 1, the model is definitely superior with respect to the other solutions in the case of complex images, such as the Div2K ones. where the other models are not capable enough in keeping finer details or high-frequency contents.

4. Conclusions

In this work we try to find out some insights regarding a custom version of the SIREN network, presented in [8], by

[2] https://gist.github.com/alper111/8233cdb0414b4cb5853f2f730ab95a49

[3] https://github.com/twhui/SRGAN-PyTorch

Dataset		Metric	SIREN MSE	SIREN VGG	SIREN P.E.	ReLU P.E.	Bicubic	SRGAN
Skimage	×2	PSNR	31.817	32.387	33.140	22.461	**33.203**	28.370
		SSIM	0.911	0.936	0.938	0.395	0.956	**0.957**
	×4	PSNR	26.647	**26.718**	26.396	23.358	26.667	24.673
		SSIM	0.825	0.831	0.804	0.548	**0.837**	0.770
	×8	PSNR	22.658	22.636	22.447	21.745	22.875	**23.698**
		SSIM	0.704	0.704	0.675	0.582	0.703	**0.723**
Div2K	×2	PSNR	29.502	29.000	29.888	23.608	30.607	**31.271**
		SSIM	0.893	0.886	0.895	0.512	0.920	**0.943**
	×4	PSNR	24.590	24.576	24.397	22.812	25.490	**27.067**
		SSIM	0.747	0.750	0.714	0.544	0.766	**0.808**
	×8	PSNR	20.908	20.952	20.732	20.387	21.744	**23.672**
		SSIM	0.592	0.591	0.561	0.504	0.601	**0.656**

Table 1. PSNR and SSIM metrics comparison between different networks. For the *Skimage* dataset the *Camera* image is used while for the *Div2K* dataset the 49th image is employed. ×2, ×4 and ×8 are the scaling factors. SIREN and ReLU have 3 hidden layers with 512 features each. The training procedure consists of 500 iterations. As mentioned, in simple images, like *Camera*, all the methods are almost comparable, and our SIREN implementation with VGG loss achieves the best results when using a ×4 factor. For complex *Div2K* images instead SRGAN network outperforms the other models both for PSNR and SSIM metrics.

comparing it with some usual MLP implementations and by applying some ablation studies on our model. We show how important is the idea of the initialization scheme in the field of sinusoidal non-linearities and how this kind of functions can be exploited when dealing with simple image fitting or Poisson image reconstruction tasks. The last goal of this work is related to the exploitation of a SIREN-based network for the Single Image Super-Resolution task. This is considered as one of the most challenging problems in computer vision and a lot of dedicated studies have been proposed in order to fulfill it. We succeed in applying a relatively simple network (SIREN is just a MLP with some custom solutions) on this task with outcomes that can be considered competitive in some areas with the most renowned solutions. By challenging this solution in the field of super resolution, we show how the concept of implicit representation of signals can be exploited in a way that is much better with respect to simple interpolation algorithms. We also recognize the limits of this kind of networks, that are related to the fact that the actual implicit neural representation does not retain enough details and high-frequency elements within some complex images. The results in the case of important scaling factors (such as ×4 and, mostly, ×8) lack exactly in the higher-frequency details, suggesting that a sort of hybrid-solution between a Generative model and an implicit neural representation solution could lead to better performances.

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. 1, 13

[2] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In Y. W. Teh and M. Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. JMLR Workshop and Conference Proceedings. 4

[3] R. Keys. Cubic convolution interpolation for digital image processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 29(6):1153–1160, 1981. 3

[4] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4681–4690, 2017. 2, 3, 7

[5] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001. 1, 11

[6] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020. 2, 7, 12

[7] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. 3

[8] V. Sitzmann, J. N. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein. Implicit neural representations with periodic activation functions. In *arXiv*, 2020. 1, 2, 3, 4, 5, 7, 9, 10, 12

[9] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and the scikit-image contributors. scikit-image: image processing in Python. *PeerJ*, 2:e453, 6 2014. 1

[10] W. Yang, X. Zhang, Y. Tian, W. Wang, J.-H. Xue, and Q. Liao. Deep learning for single image super-resolution: A brief review. *IEEE Transactions on Multimedia*, 21(12):3106–3121, Dec 2019. 1, 2, 3

S1. Supplementary Material

We report in this section additional material in order to better understand the quality of our research.
Subsection S1.1 shows more in detail the comparison between our network and a ReLU-based MLP. Subsection S1.2 focuses on the comparison between activation distributions obtained by changing the initialization scheme of the SIREN network, whereas subsection S1.3 emphasises the differences between different models trying to accomplish the SISR task.

S1.1. Comparison with ReLU-based MLP

Fig. S1 depicts PSNR values' trends for the two compared networks: despite a greater variance as the steps number grows, SIREN shows better performances both from a qualitative point of view and from the perspective of convergence speed within a finite number of steps.

S1.2. Ablation Studies

In the following we report some supplementary details regarding the ablation studies we conduct, aimed at showing the importance of a good initialization scheme for the effective training of the SIREN network. By changing the first and hidden layers' weights and ω_0 factor from the ones provided in [8], we are able to face up different situations, striving to acknowledge the strength of each of these terms in the overall architecture.

Initialization of first layer's weights As explained in section 3.3.1 of the report, the key idea of the initialization scheme is to preserve the distribution of activations through the network so that the final output at initialization does not depend on the number of layers. Convergence speed and accuracy can be significantly influenced by a wrong distribution of the weights in the first layer of a SIREN.
Fig. S3 shows the comparison between the initialization scheme proposed by Sitzmann et al., the Kaiming uniform and the Xavier uniform distributions, applied to the first layer. As we can see, since the Kaiming and Xavier uniform distribution have bounds similar to the ones in the first layer of the standard SIREN, the results are not showing so much differences.

Initialization of hidden layers' weights Section 3.3.2 of the report describes that, when each component of the weights matrix is uniformly distributed, the output of each single layer should converge to a normal distribution. Fig. S4 shows again the comparison between the initialization scheme proposed in [8], the Kaiming uniform and the Xavier uniform distributions, this time applied to the hidden layers. In this case the difference between the recommended scheme and the other ones is explicit: we can observe how, when using different initialization schemes, the dot product between the weights and the input values fails in recreating a Normal distribution.
This is reflected also in the activation spectrum, which shows a much more unstable trend than the standard one. This is due to the fact that the bounds of the hidden layers' weights distribution are not tailored on the presence of the ω_0 factor as they are on the distribution suggested by Sitzmann et al.

Initialization of ω_0 As reported in section 3.3.3 of the report, this parameter is fundamental for increasing the spatial frequency of the first layer, in order to better match the frequency spectrum of the signal, and for accelerating the training of the model by parameterizing the weight matrix.
As also mentioned in the previous section, the presence of the factor ω_0 equal to 30 (as in [8]) has a negative effect when trying to change the initialization scheme of the weights of the hidden layers of the network.
Fig. S2 demonstrates that choosing $\omega_0 = 1$ for the hidden layers leads to PSNR values trends, based on different initialization schemes, that are dissimilar from the behaviour shown in Fig. 4b of the paper. At the cost of losing the gradients boost, we manage to obtain *non-flat* curves for all the proposed schemes.

S1.3. SISR task

In this section we present some baselines approaches and their results when applied on several common SISR tasks. Then we compare the obtained results with some variants of the original SIREN model, in order to understand whether and how it can be improved in this sense.

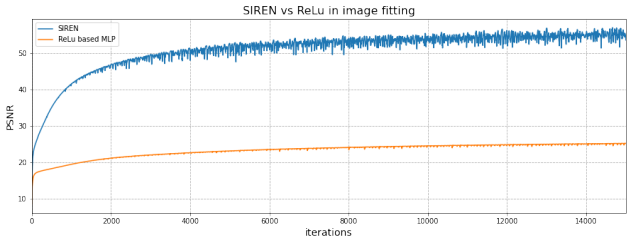

Figure S1. Comparison of the PSNR values trends on 15,000 iterations, based on our implementation of the SIREN and on a ReLU-based MLP. Fitting task is performed over the *Camera* image.

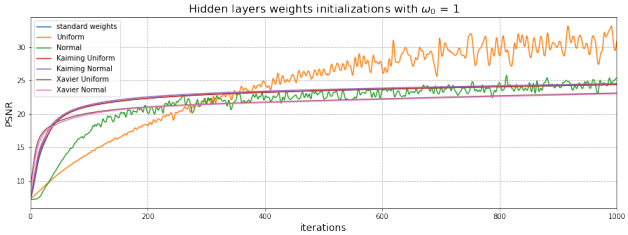

Figure S2. Comparison of PSNR values trends on 1,000 iterations, obtained by using different initialization schemes for the hidden layers' weights and $\omega_0 = 1$. SIREN model has 3 hidden layers.

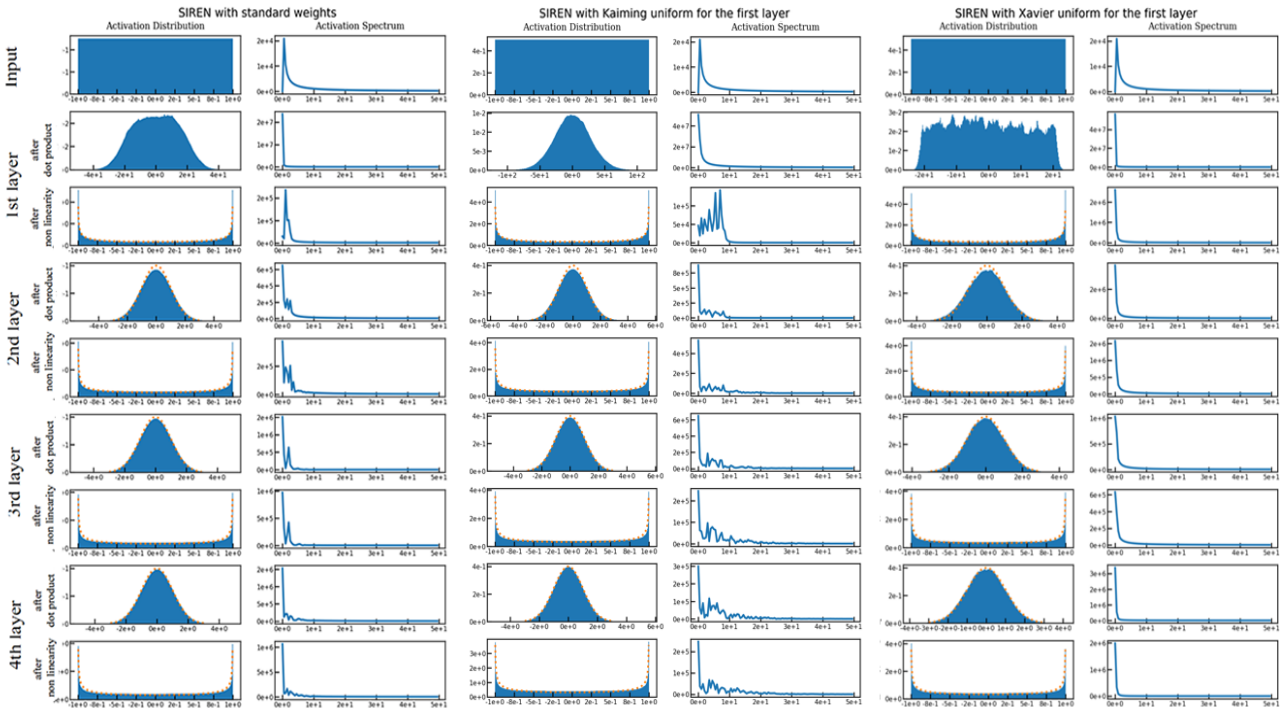

Figure S3. Activation distribution and spectra with [8], Kaiming uniform and Xavier uniform initialization schemes applied to the first layer. Increasing layers from top to bottom. Orange dotted line visualizes analytically predicted distributions.
As previously described, there are no substantial differences between the effects of the initialization schemes.

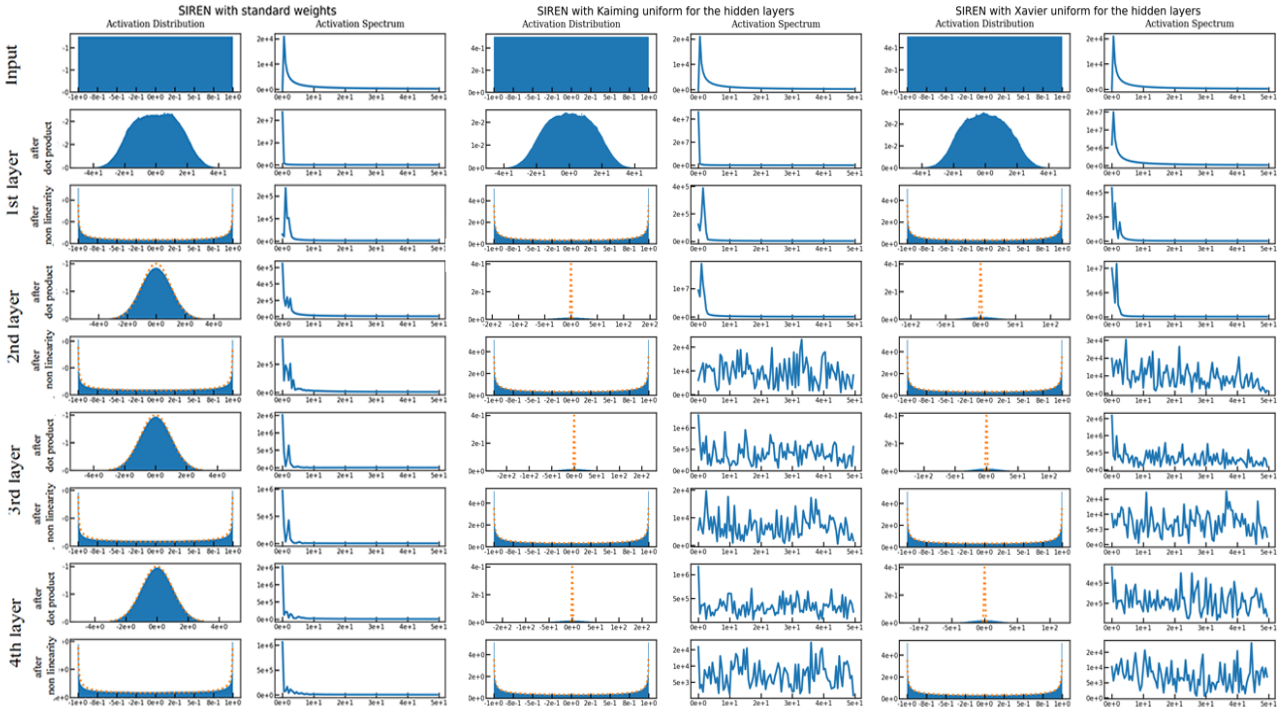

Figure S4. Activation distribution and spectra with [8], Kaiming uniform and Xavier uniform initialization schemes applied to the hidden layers. Increasing layers from top to bottom. Orange dotted line visualizes analytically predicted distributions.
As previously described, the difference between the several schemes is more evident, showing an unstable activation spectrum for both the Kaiming and the Xavier uniform distributions.

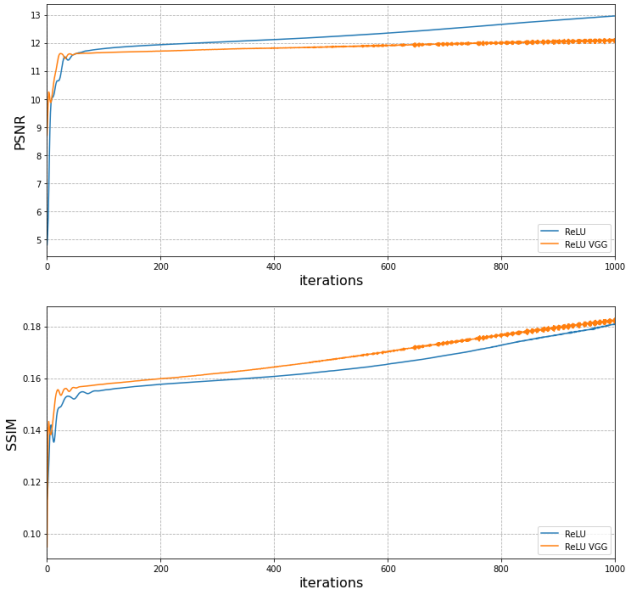

Figure S5. Comparison between standard ReLU and ReLU with VGG (3 hidden layers, 256 features each) on a SISR task over image 143 from Div2K. Training on 1,000 steps, LR 1×10^{-4}.
ReLU with MSE loss reaches a higher PSNR with respect to the model using VGG loss (12.96 and 12.00, respectively), while SSIM values are very close (≈ 0.18 both).

S1.3.1 Bicubic interpolation

The first and most used methodology that we describe as baseline is the *Bicubic interpolation*. Specifically, in this method the low-resolution image taken as Numpy array is first of all transformed to a *PIL image*, then it is subjected to a pytorch resize by using as filter `PIL.Image.BICUBIC`.
As reported in the main paper (section 3.4.2), some of the values related to the metrics are comparable (if not superior) to the ones of the other models that we used.
For what concerns the corresponding visual feedback, Fig. S9 shows how the Bicubic interpolation approach manages to get comparable and perceptibly appreciable results.

S1.3.2 SRGAN

As described in the report, we used a pretrained model [4] in a black-box way. Specifically, the provided configuration file for the testing on the SRGAN model has been modified [5] so that it refers to a *"fake"* dataset that has the same folder structure as [5] and it is filled with the images we are interested in.
As mentioned in the report, the exploited *test mode* performs downsampling on the high-resolution ground-truth provided as input and, starting from this low-resolution version, the super resolution task is accomplished. We have to specify that the model that we used is only compatible with

[4] https://github.com/twhui/SRGAN-PyTorch
[5] https://github.com/paologastaldi-polito/siren/blob/master/src/srgan/config_test.json

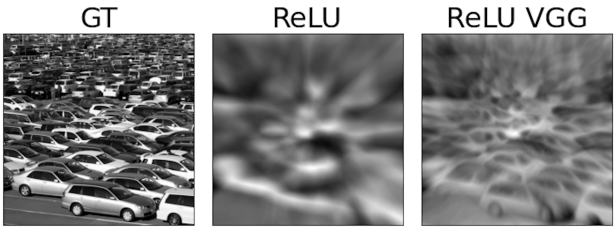

Figure S6. Same task and same models of Fig. S5. Even if the fitting process shows similar PSNR and SSIM values, visual results are quite different. By adopting the VGG loss we can find out the subject on the image and its borders.

$\times2$, $\times4$ and $\times8$ scaling factors.
In order to guarantee the fairness of the comparison we made sure that the inner downsampling transformation is compatible with the ones that we operated on the input images for every other model experimented. In fact, the actual transformation (dependent on the parameter `midcrop_size`) is performed with a reimplemented version of `torchvision.transforms.Resize`, therefore compatible with our implementation.
As we can notice from Fig. S9, SRGAN performs definitely better in terms of sharpness, radiance and contrast with respect to the other models.

S1.3.3 Applying SIREN to SISR

In this section we analyze how the implementation of some changes to the standard SIREN can influence the performance on addressing a SISR task.
The overall behaviours and trends we introduce are summed up in Table S1. In order to have a visual feedback of what is described below please refer to Fig. S7 and Fig. S9.

With MSE loss The results of applying a simple MSE loss show a modest behavior for small scaling factors (such as $\times2$ or $\times4$), but it finds some difficulties with high frequency detailed images. Being the MSE applied in a pixelwise manner, even if it enables the overall architecture to create an implicit representation of the input, it cannot perceive complex relations between pixels in the content, so its performance in SISR is surpassed by solutions that take the details and high-frequency contents representation as their principal goal. This is why, in the following, we try to enhance the capability of the usual implicit neural representation by implementing some changes both in the loss function and in the way pixels are sampled from the provided grid.
As we can see from Fig. S8, the standard implementation behaves quite well both in terms of PSNR and SSIM as the steps grow, but it is outperformed by some of its variants in the transient part of the training process.
We can also observe the visual comparison between the MSE-based version of SIREN and the same version of a ReLU-based MLP in Fig. S7. As we can see, SIREN natu-

SIREN	Description
MSE Loss	PSNR increased with many hidden features (512 or more), SSIM almost stable Good performances with $\times 2$ and $\times 4$ scaling factors High frequency content not well retained and represented in SISR
VGG Loss	Stability increased with many hidden layers, PSNR comparable with MSE loss but higher SSIM values Slight improvement with a small number of iterations No improvements as the number of steps grows
P.E.	Boosted convergence speed with few hidden layers (2 or 3) Earlier degeneration with many hidden layers (5 or more), PSNR values swings in a wider range Faster growth of the PSNR and SSIM during the early stages of the training process Penalized maximum PSNR and SSIM values reachable

Table S1. Summary of our different approaches to the SIREN model while performing a SISR task.

rally produces better results, due to the usage of well tuned solutions such as sine non-linearities, mentioned in the report.

With Positional Encoding as in NeRF As already described in section 3.4.1 of the main paper, we try to apply the positional encoding presented in [6] in order to be able to better represent finer and higher-frequency details.
By observing Fig. S8 it is evident how the so called "NeRF mode" (the positional encoding technique provided by [8]) enables the model to reach higher PSNR and SSIM values in a minor number of steps with respect to all the other SIREN variants. However, taking into account both metrics, the introduction of positional encoding slightly penalizes the maximum reachable values.
Also in this case we can refer to Fig. S7 in order to make a visual comparison between the two different MLP implementations described above. Firstly, we can observe a slight deterioration of the image produced by our SIREN model equipped with positional encoding with respect to the standard one. We can state the opposite for what concerns the ReLU-based MLP: it definitely benefited from the introduction of the described technique and, visually, it almost reaches the performance of the respective SIREN implementation.

With VGG loss The last approach presented involves the usage of a VGG loss (9), instead of the classic MSE one. We applied this technique on a SIREN, based on the good results obtained on images with high-frequency contents even by using ReLU-based MLPs.
Fig. S6 allows us to compare results generated with MSE and VGG loss on a "complex" image taken from DIV2K dataset. It is clear that the introduction of the VGG loss enables the model to represent, at least, some of the outlines that are showed in the ground-truth image, with respect to the standard ReLU-based MLP that is not even able to make the idea of the content. The same improvement as the visual one is not present in the PSNR and SSIM trends, shown in Fig. S5.
By observing Fig. S8 we understand that, for what concerns the SIREN implementation with the VGG loss, it benefits a bit with a small number of iterations, but it does not improve as the steps grow. However, if we visually refer to Fig. S7, it is evident how the same improvement that we can observe between the ReLU-based variants is not visible in the respective SIREN variants.
In this SIREN implementation, the VGG loss is inside the network even if its VGG parameters are already pre-trained (we do not re-train them, we just use them). This decision is based on the need of swapping the model position between CPU and GPU devices in an easier way.

With mixed approach Starting from the results obtained with P.E. and the VGG loss, we consider to apply them both to our model. At first we mixed these approaches on a ReLU network. The results (that can be observed in Fig. S8a) show how the ReLU-based MLP with mixed approaches, after some slight difficulties in the initial phases, tends to reach better results as the number of iterations grows.
Unfortunately, this benefit seems not to appear in the SIREN performance at all. On the contrary, as shown in Fig. S8b PSNR and SSIM results are definitely worse than the other ones in terms of maximum values and convergence speed.
By considering a visual comparison, Fig. S7 shows how SIREN behaves much worse than its previous runs. The ReLU-based MLP, on the other hand, seems to generate less noise with respect to the previous approaches.
In general, SIREN seems to be more independent from the approach applied. As shown in graphs, the classic network can reach results that are comparable with those obtained with Positional Encoding and VGG loss techniques. Only the combination of the two approaches seems to negatively affect the result.

S1.3.4 Model Comparison

At each step of the fitting process for the SISR task we let the model predict also a temporal HR image in order to compare it with the HR ground-truth and generate the PSNR and SSIM graphs showed in Fig. S8. This means that the represented trends are the result of metrics applied

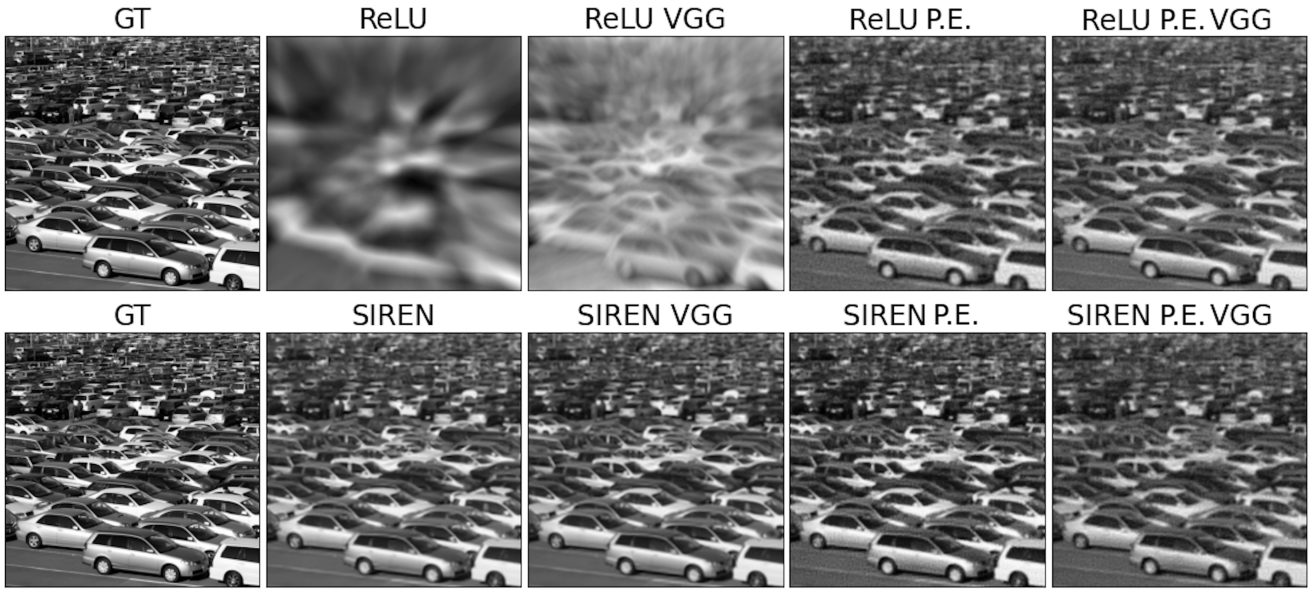

Figure S7. Visual comparison over the reconstruction of image 143 of *Div2K* from a 128×128 to a 512×512 resolution with SIREN and ReLU-based networks and their variants, with 3 hidden layers and 256 features each. Training on 1,000 steps, LR 1×10^{-4}.

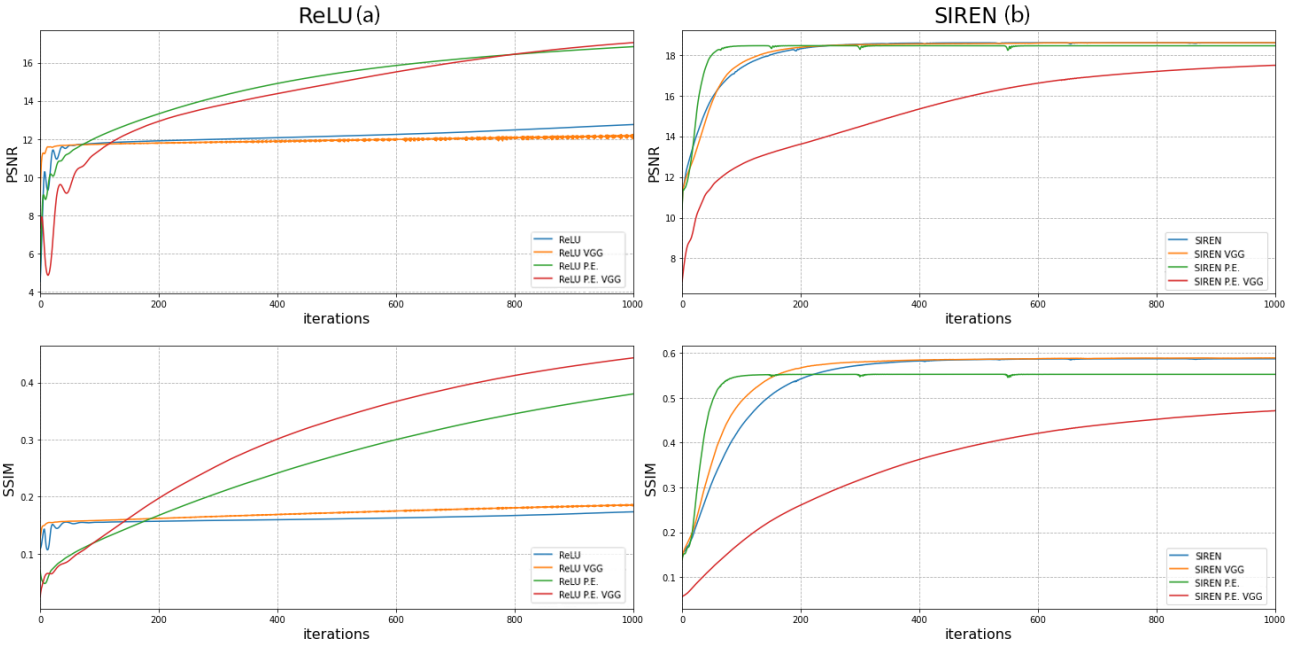

Figure S8. Metrics comparison over the reconstruction of image 143 of *Div2K* from a 128×128 to a 512×512 resolution with SIREN and ReLU-based networks and their variants, with 3 hidden layers and 256 features each. Training on 1,000 steps, LR 1×10^{-4}.

on the super resolution computations (SISR) and not on the fitting temporal results.

S1.3.5 DIV2K datasets usage

In this section we want to highlight how we extracted images from the DIV2K dataset [1] by keeping them compatible with the format of the variants implemented for the purpose of this work. We implemented a Dataloader [6] that enabled us to download and extract images as we need. Starting from some complex images extracted from this dataset we show in Fig. S9 some extra demonstrations of how some different models behave on these samples.

[6] https://github.com/paologastaldi-polito/DIV2K-loader

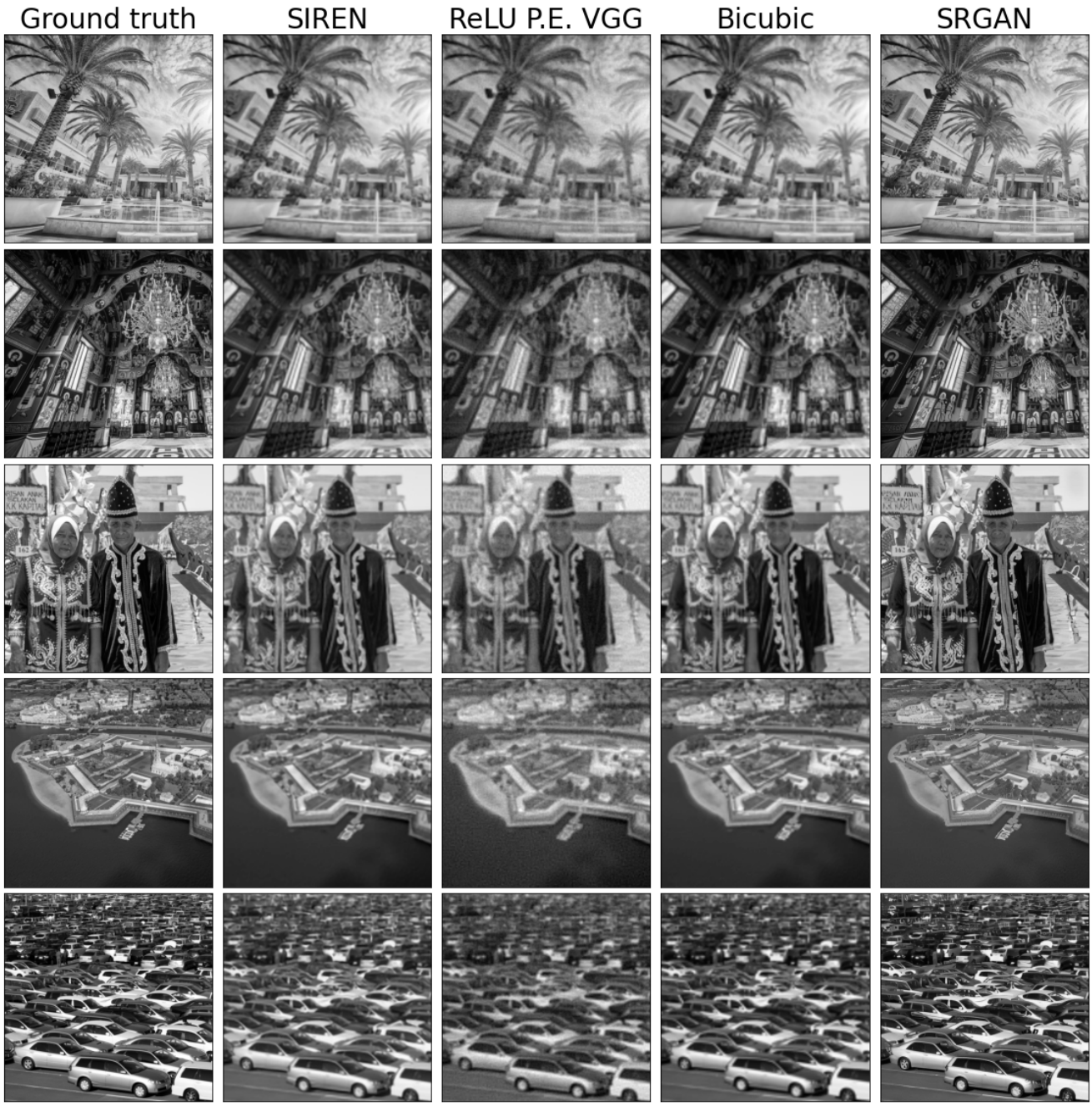

Figure S9. Visual comparison over the reconstruction of several images of *Div2K* dataset (respectively, indexed 2, 28, 68, 82, 143) from a 128×128 to a 512×512 resolution. Training consists of 500 steps. SIREN and ReLU have 3 hidden layers with 512 features each.