IMPLICIT NEURAL REPRESENTATION WITH PERIODIC ACTIVATION FUNCTIONS

Artificial Intelligence and Machine Learning
Politecnico di Torino

Matteo Abbamonte
Paolo Gastaldi
Stefano Gennero
Alkis Koudounas

OUTLINE

- The importance of Implicit Neural Representation
- SIREN innovative approach
- Datasets and Metrics
- Experiments
  - Propose a 2 hidden layers SIREN implementation to fulfill several known tasks such as Image Fitting and Image Reconstruction
  - Compare the presented network with a ReLU-Based MLP
  - Ablation Studies over the initialization schemes
  - Single Image Super-Resolution task

IMPLICIT NEURAL REPRESENTATION

A novel way to parameterize signals of all kinds

Implicit Neural Representations parameterize a signal as a continuous function that maps the domain of the signal to whatever is at that coordinate.

These models are not coupled to spatial resolution, so the memory required to parameterize the signal scales only with the complexity of the underlying signal itself.

They can be sampled at arbitrary spatial resolution, and this is useful for several applications, such as super resolution or parameterizing signal in high dimensions.

SIREN

The power of
periodic activation functions

Being Φ a class of functions that satisfies equations of the form

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

Proposal of a neural network architecture that uses the sine as periodic activation function:

$$\Phi(x) = W_n(\phi_{n-1} \circ \phi_{n-2} \circ ... \circ \phi_0)(x) + b_n$$

$x_i \mapsto \Phi(x_i) = sin(W_i x_i + b_i)$: *Sine activation function*

$\phi_i : \mathbb{R}^{M_i} \mapsto \mathbb{R}^{N_i}$: *i^{th} layer of the network*

$W_i \in \mathbb{R}^{N_i \times M_i}$: *Weight matrix*

$b_i \in \mathbb{R}^{N_i}$: *Biases*

$x_i \in \mathbb{R}^{M_i}$: *Input*

SIREN - Strengths

- Any derivative of a SIREN... *is a SIREN itself!*
  - Good behavior in representing signals and their derivatives
  - Poisson image reconstruction is possible
- Distribution of activations preserved through the whole network
  - No exploding or vanishing gradients
- Better matching of the frequency spectrum of the signal by introducing ω_0 in the first layer
- Training acceleration by using ω_0 also in the hidden layers

SIREN - Initialization scheme

- First layer weights distribution: $W \sim \mathcal{U}\left(-\sqrt{\frac{6}{fan_{in}}}, \sqrt{\frac{6}{fan_{in}}}\right)$
- Factorized inner weight matrix: $W = \hat{W} * \omega_0,\ \hat{W} \sim \mathcal{U}\left(-\sqrt{\frac{6}{\omega_0^2 fan_{in}}}, \sqrt{\frac{6}{\omega_0^2 fan_{in}}}\right)$
- Input to the sine activation function is *Gaussian Normal distributed*
- Output of the sine activation function is ~ *Arcsine distributed*

DATASETS

Skimage: Collection of algorithms for image processing that also contains a data directory with test images

- Camera image

BSDS500: 500 natural images, ground-truth human annotations and benchmarking code

- Starfish image (center-cropped to 321x321 and resized to 256x256)

Div2K: 1000 2K resolution images divided into 800 images for training, 100 for validation and 100 for testing.

- Images indexed 2, 28, 49, 68, 82, 143

METRICS: PSNR and SSIM

PSNR: *Peak Signal-to-Noise Ratio*

- *dB* scale
- Pixelwise metric
- Good results with MSE loss (estimate the absolute error)

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

SSIM: Structural Similarity Index Measure

- Based on image distortion
- Reflects in a better way what the human eye perceives

$$SSIM = \frac{2\mu_x \mu_{\hat{x}} + k_1}{\mu_x^2 + \mu_{\hat{x}}^2 + k_1} \cdot \frac{\sigma_{x\hat{x}} + k_2}{\sigma_x^2 + \sigma_{\hat{x}}^2 + k_2},$$

$$where:$$

$$\mu = mean, \sigma = covariance, \sigma^2 = variance$$

$$k_1, k_2 = constant\ relaxation\ terms$$

EXPERIMENTS

Our work

Our SIREN Architecture

- Linear and sine activation function as first layer
- Linear and sine activation function as hidden layers
- Linear block as final layer
- ω_0 and weights distribution are the recommended ones

Input: grid of values in [-1, 1] with sidelength S

Output: image with a sidelength of S

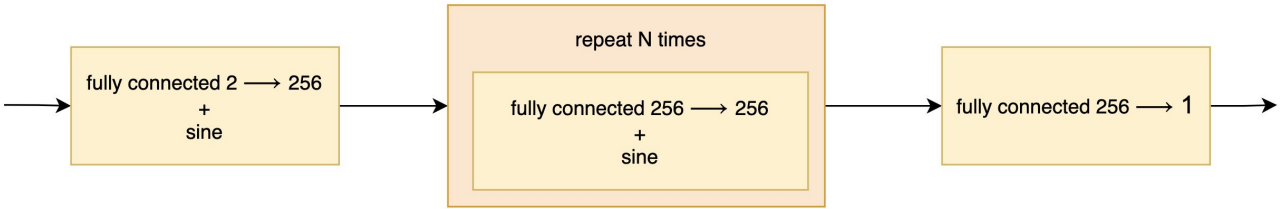


Image Fitting

Given

- $\mathbf{x}_i = (x_i, y_i)$: pixel coordinate
- $f(\mathbf{x}_i)$: associated RGB colors

Image Fitting consists in finding the function $\Phi : \mathbb{R}^2 \mapsto \mathbb{R}^3, x \mapsto \Phi(x)$ that parameterizes a given discrete f in a continuous fashion and outputs image colors at pixel coordinates, solely depending on Φ and $f(\mathbf{x}_i)$.

DEFINITION: MSE Loss

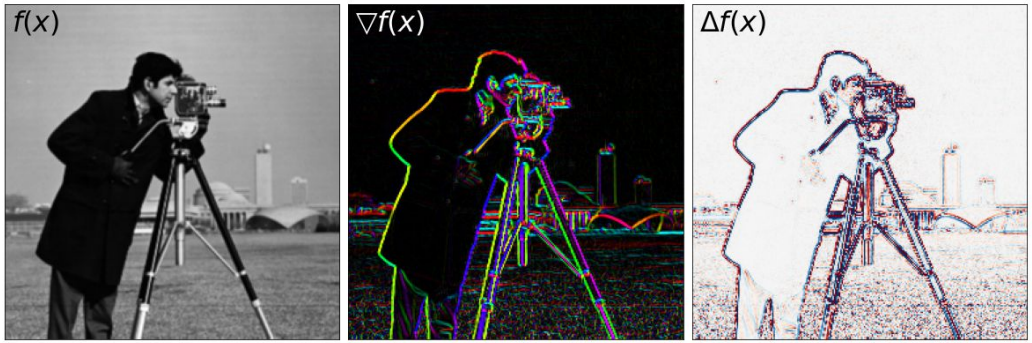
$$MSE = \frac{\sum_{i=0}^{D}(x_i - \hat{x}_i)^2}{D}, \; D = number\ of\ pixels$$

Measures the average squared difference between the estimated values and the actual ones in a pixelwise manner.

Tends to achieve high PSNR values.

Supervision on Image

- MSE loss between the predicted image and the target ground-truth image
- Camera image from the Skimage dataset
- Both Gradients and Laplacian are also visualized



SIREN with 2 hidden layers, 256 hidden features each. Training on 15,000 steps, LR 1e-4

Image Reconstruction

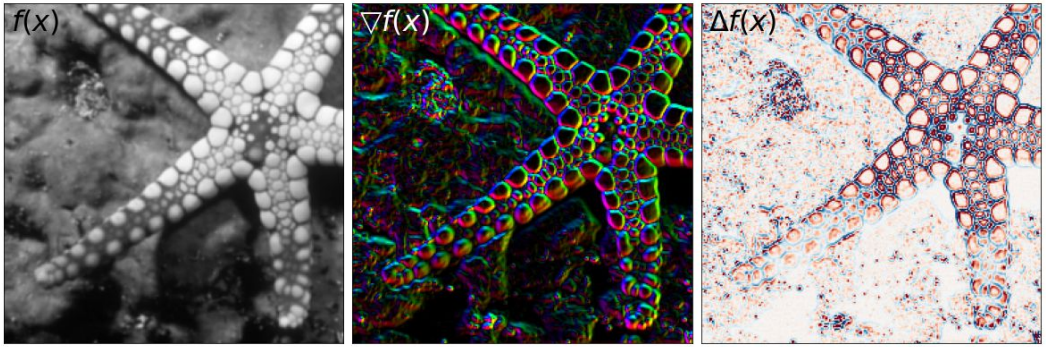
Representing an unknown ground-truth signal by using its derivatives to solve a Poisson Equation

Supervision on gradients

Implicit representation supervised on ground-truth gradients via

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

- Starfish image from BSDS500 Dataset
- Sobel Filter applied on the original image
- Gradient scaled by a x10 factor



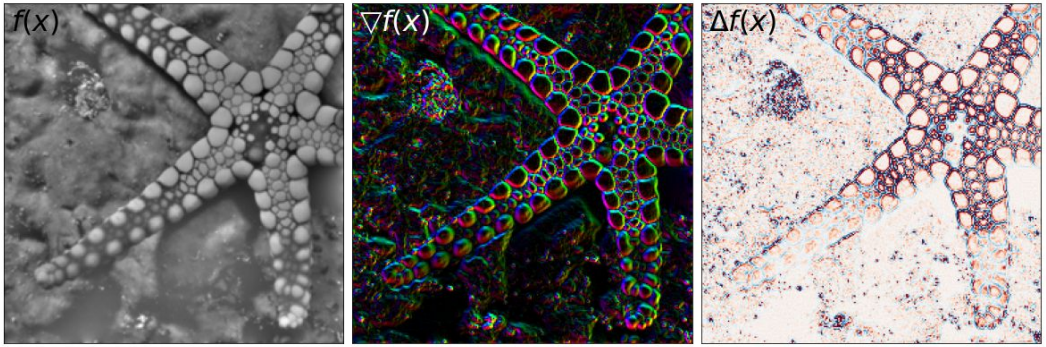
SIREN with 2 hidden layers, 256 hidden features each. Training on 15,000 steps, LR 1e-4

Supervision on laplace

Implicit representation supervised on ground-truth gradients via

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

- Starfish image from BSDS500 Dataset
- N-D Laplace Filter applied on the original image
- Laplacian scaled by a x10,000 factor



SIREN with 2 hidden layers, 256 hidden features each. Training on 15,000 steps, LR 1e-4

Comparison with ReLU

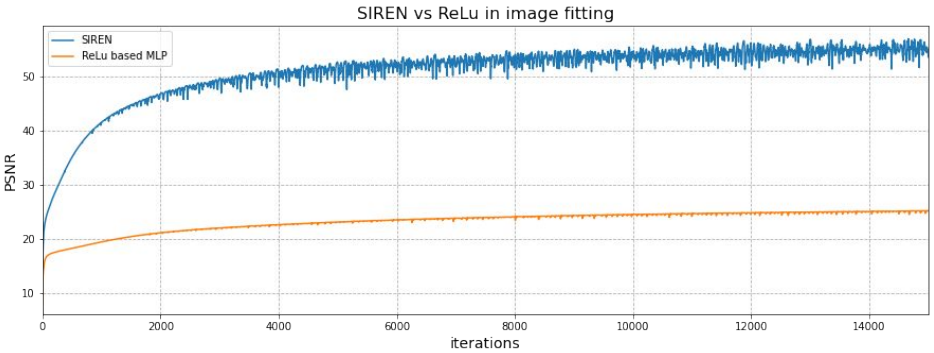
Fair comparison

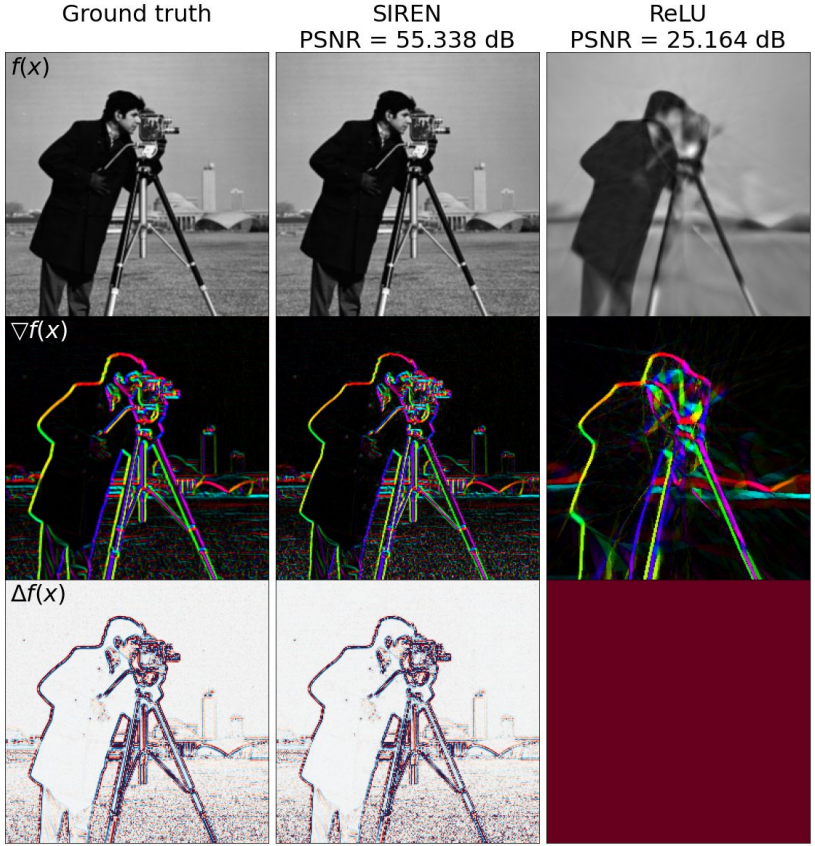
SIREN and ReLU-based MLP performance are comparable due to the following precautions:

- Same number of layers and features for both architectures
- Weight distribution initialization adapted to Kaiming Normal for ReLU-based MLP
- ADAM optimizer for both architectures
- Learning Rate 1e-4 for both architectures
- 15,000 steps training process for both architectures
- MSE used as loss function for both architectures

Image Fitting

- ReLU has lower PSNR value wrt SIREN
- ReLU produces worse visual results
- ReLU cannot reproduce the gradient faithfully
- ReLU is unable to represent second-order derivatives

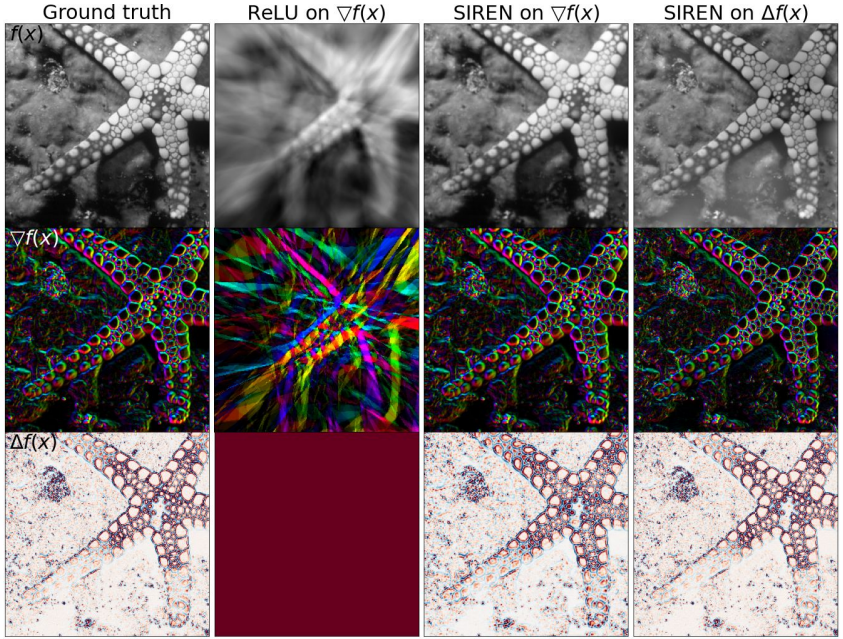




ReLU and SIREN with 2 hidden layers, 256 hidden features each. Training on 15,000 steps, LR 1e-4

Image Reconstruction

- ReLU bases the reconstruction on a poor fitting result
- ReLU produces worse visual results
- SIREN is much more prone to derivation and integration
- ReLU unable to represent second-order derivatives



ReLU and SIREN with 2 hidden layers, 256 hidden features each. Training on 15,000 steps, LR 1e-4

Ablation studies

Ablation studies - Weights initialization

Initialization schemes are pivotal in the training procedure of deep neural networks.

Goals of the ablation studies regarding the weights initialization of first and hidden layers:

1. Understand how powerful it is in getting good results (in terms of PSNR)
2. Understand how important it is in keeping the activations' distributions unchanged as the depth of the network grows

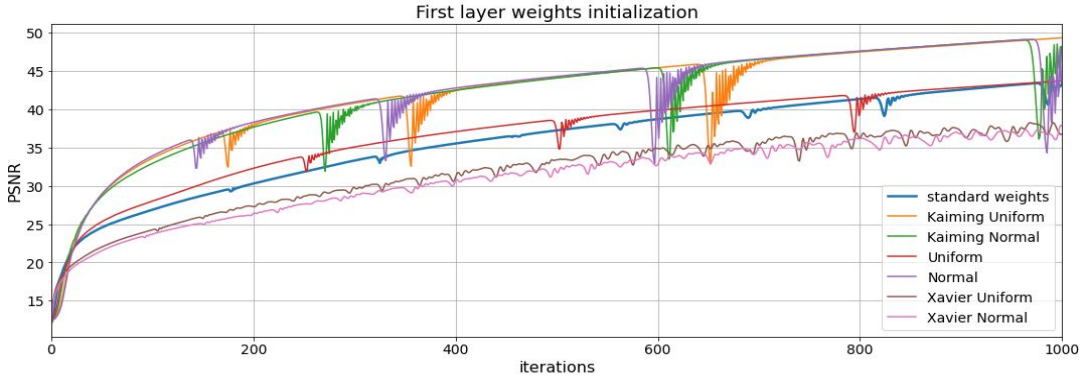
Architecture: SIREN with 3 hidden layers.

Ablation studies - First layer weights initialization

Kaiming (normal and uniform) and normal distributions
- Higher peak values, but unstable trend

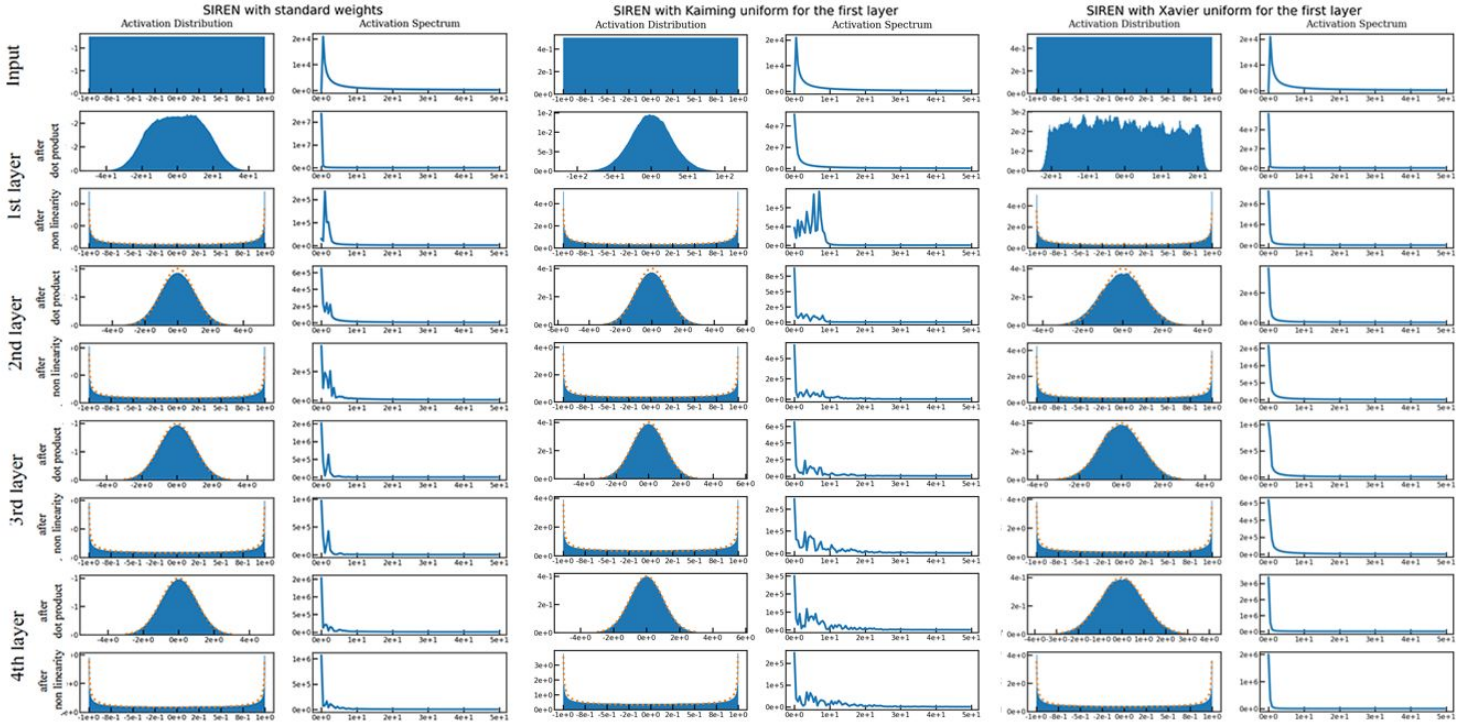
Xavier (normal and uniform) distributions
- Lower peak values, but almost stable trend

Standard SIREN and uniform distributions
- Good compromise



SIREN with 3 hidden layers, 256 hidden features each. Training on 1,000 steps, LR 1e-4

Ablation studies - First layer weights initialization

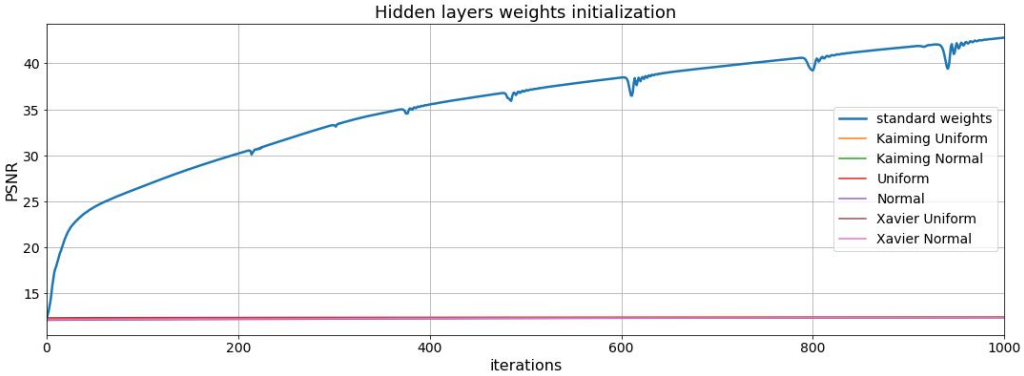


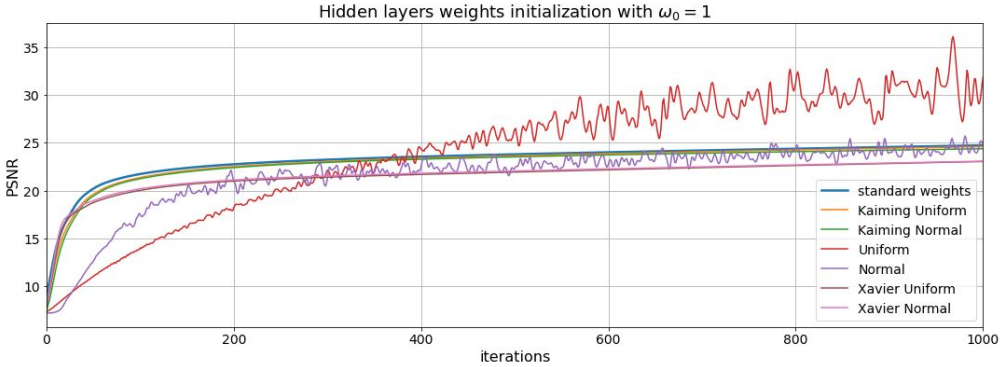
SIREN with 3 hidden layers and Standard, Kaiming uniform and Xavier uniform first layer initialization

Ablation studies - Hidden layers weights initialization

Flat curve for all the initialization schemes, except for SIREN

Behaviour due to the presence of ω_0 also in the hidden layers



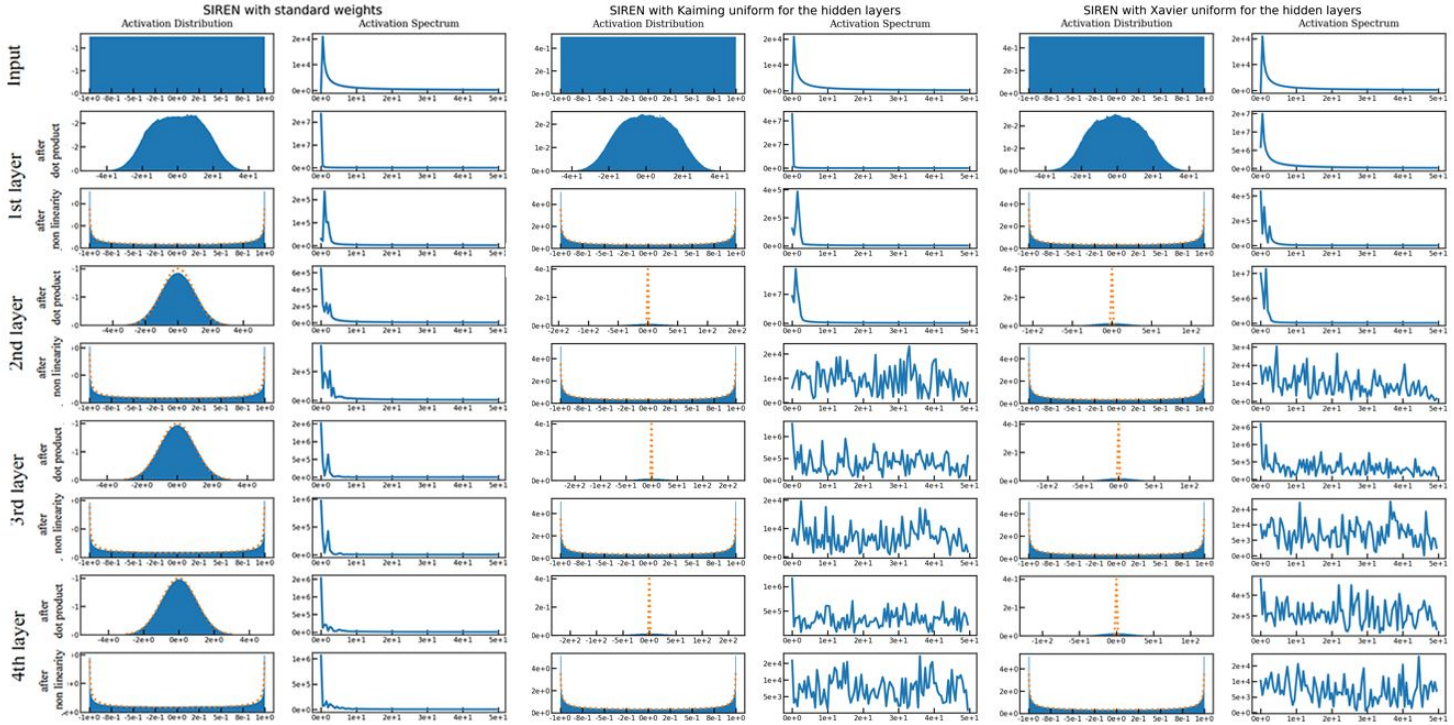


Choosing $\omega_0 = 1$ leads to *non-flat* PSNR trends for all the proposed schemes

Except for uniform (and partially normal) distribution, stable curves

SIREN with 3 hidden layers, 256 hidden features each. Training on 1,000 steps, LR 1e-4

Ablation studies - First layer weights initialization



SIREN with 3 hidden layers and Standard, Kaiming uniform and Xavier uniform hidden layers initialization

Ablation studies - ω_0 initialization

ω_0 influences the convergence of the network, but this parameter can also be used to accelerate the training phase

In general, increasing ω_0 leads to:

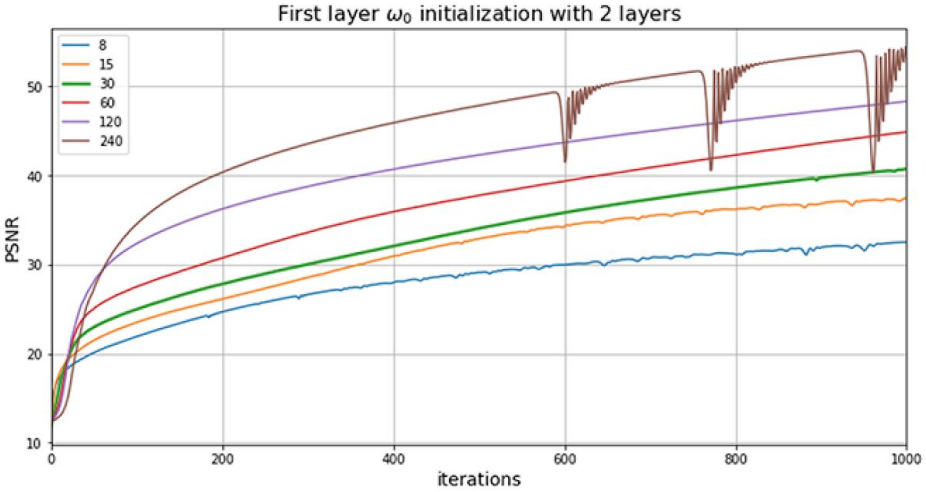
- better results (in terms of PSNR) for few iterations
- unsteady behaviour for high number of iterations
- highly unsteady behaviour for high number of layers in the network

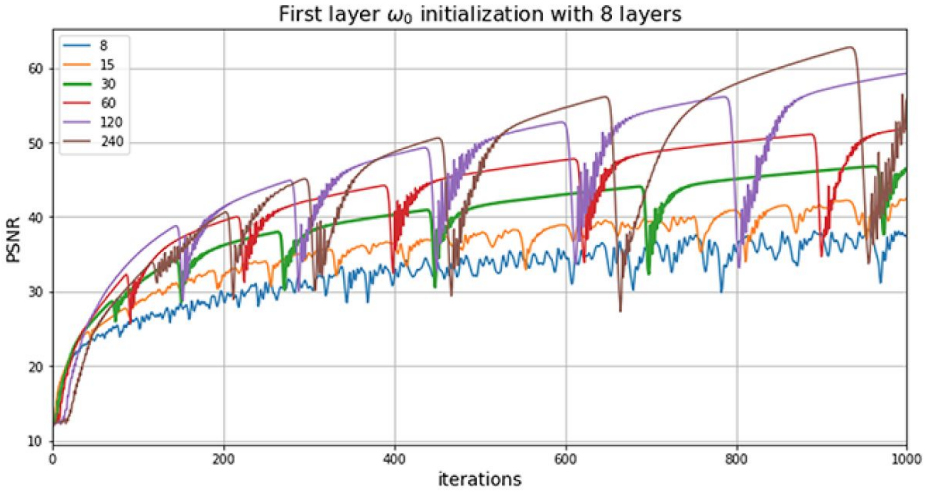
In general, decreasing ω_0 leads to:

- steady behaviour for high number of iterations and layers in the network
- poor results due to a very slow progress in training phase

Ablation studies - First layer ω_0 initialization

The influence of the first layer on the total network behaviour is quite limited, in fact the effects of high ω_0 values on the convergence are not much disruptive

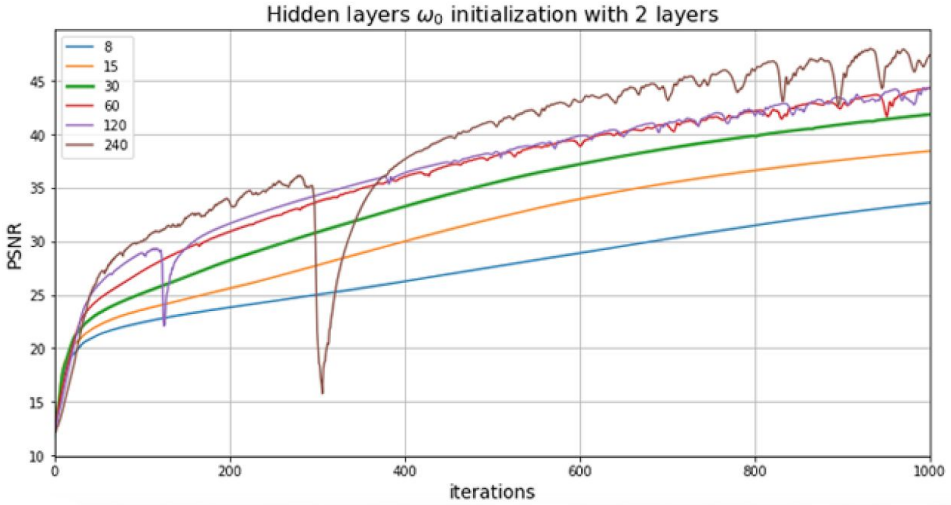


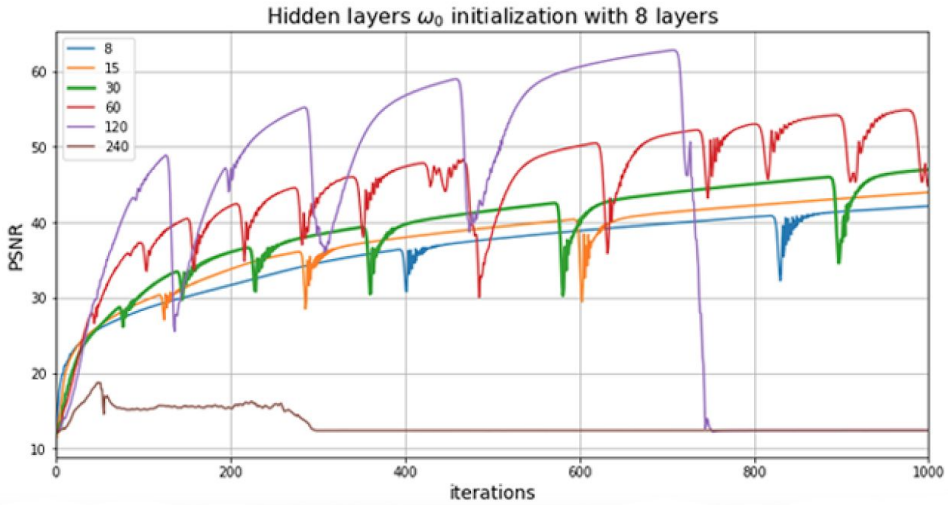


SIREN 256 hidden features each hidden layer. Training on 1,000 steps, LR 1e-4

Ablation studies - Hidden layers ω_0 initialization

The hidden layers have a high influence on the overall network behaviour, in fact the effects of high ω_0 values on the convergence are strong





SIREN 256 hidden features each hidden layer. Training on 1,000 steps, LR 1e-4

SISR

Single Image Super-Resolution

Typical SISR Framework

SISR refers to the task of reconstructing high-resolution images from one or more low-resolution observations of the same scene.

$$y = (x \otimes k) \downarrow_s + n$$

y : *low-resolution image*

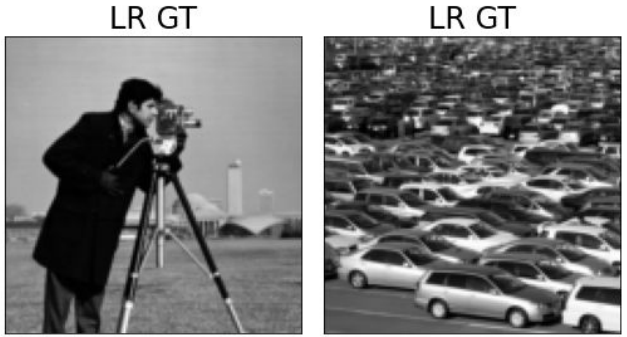
$x \otimes k$: *convolution between blurry kernel k and unknown high-resolution image x*

$\downarrow_s$: *downsampling operator with scale 's'*

n : *independent noise term*

Experiments

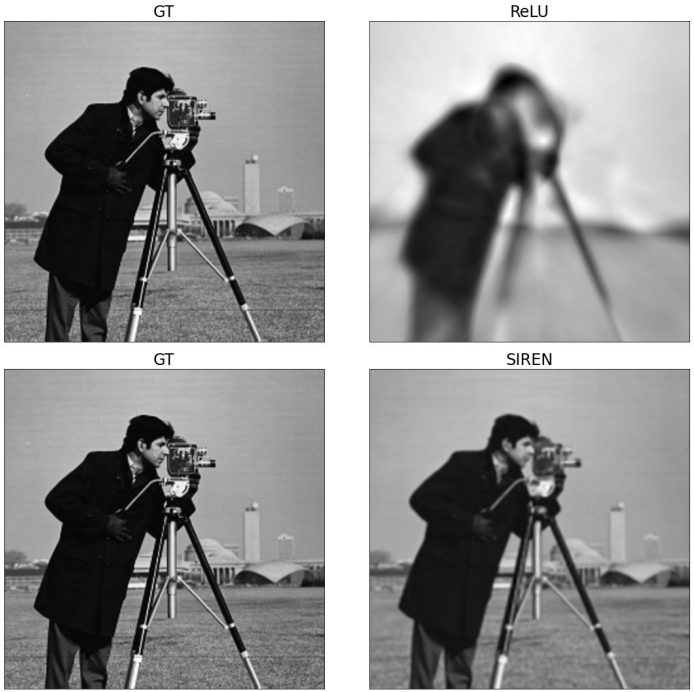
- Network with 3 hidden layers, 512 features each
- Training on 1,000 steps
- ADAM optimizer
- Learning Rate 1e-4
- Super image X4 scale (from a 128x128 to a 512x512 resolution)



Skimage camera and Div2K 143 with 128x128 resolution

MSE loss results

- Standard version of the models, used as base for the following studies
- MSE loss encounters problems when facing with higher scaling factors or higher frequency detailed images
- SIREN results are quite acceptable, while ReLU performs much worse



ReLU and SIREN with 3 hidden layers, 512 hidden features each. Training on 1,000 steps, LR 1e-4

VGG loss

Firstly introduced by SRGAN, it is an alternative to pixelwise losses.

- More details about SRGAN later on

Based on the ReLU activation layers of the pre-trained VGG19, it defines a euclidean distance between the feature representations of a reconstructed image and the reference high-resolution one.

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

I^{HR} : *reference high-resolution image*

$G_{\theta_G}(I^{LR})$: *reconstructed image*

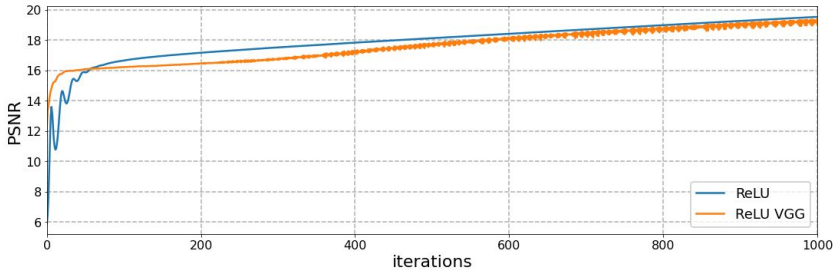
$\phi_{i,j}$: *feature map obtained by the j^{th} convolution before the i^{th} maxpooling layer within the VGG19*

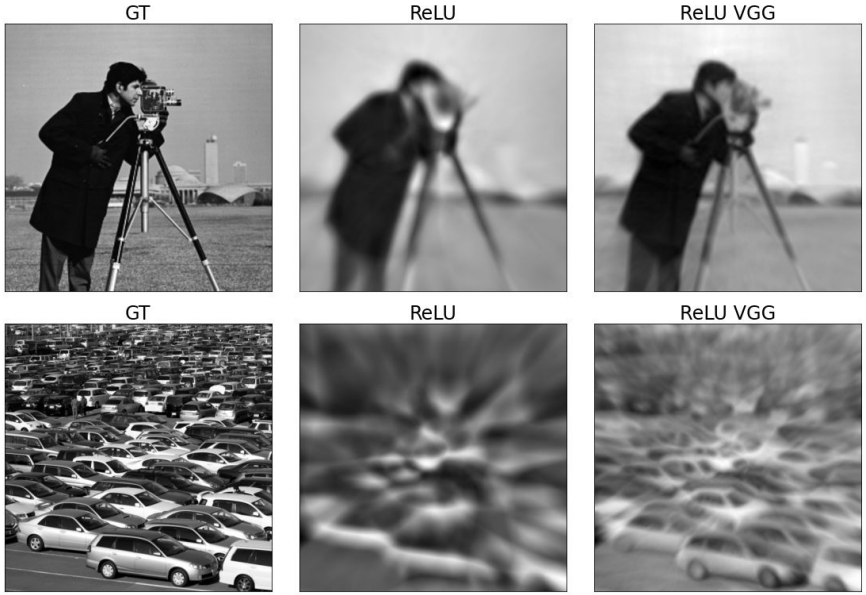
$W_{i,j}$ $H_{i,j}$: *dimensions of each feature map*

VGG loss results on ReLU

Better visual details

Similar PSNR and SSIM





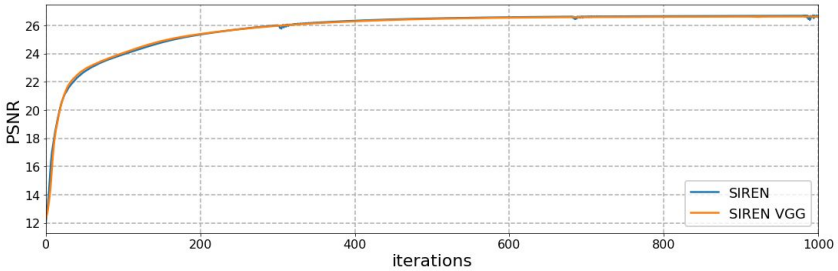
Skimage camera PSNR and SSIM

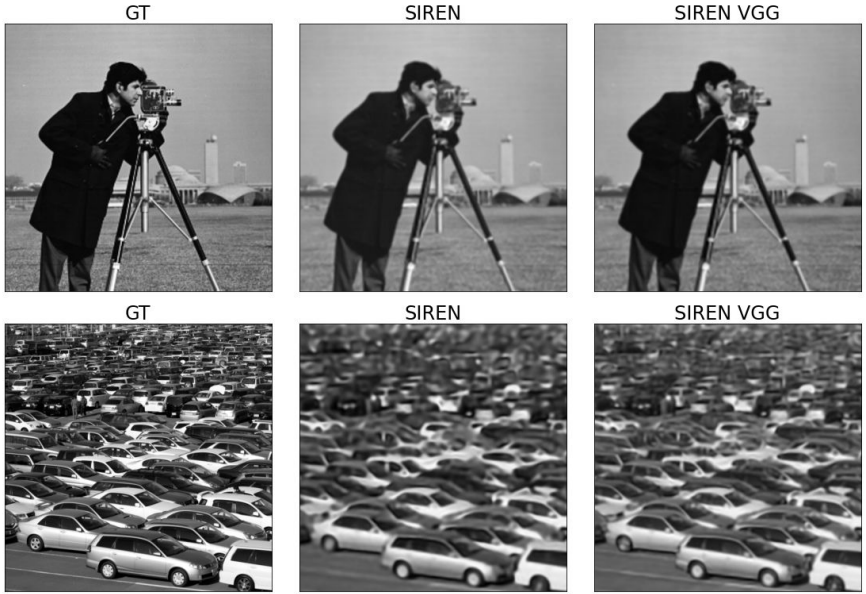
SIREN with 3 hidden layers, 512 hidden features each. Training on 1,000 steps, LR 1e-4

VGG loss results on SIREN

Slightly better visual results

Similar convergence speed, PSNR and SSIM





Skimage camera PSNR and SSIM

SIREN with 3 hidden layers, 512 hidden features each. Training on 1,000 steps, LR 1e-4

NeRF Positional Encoding (P.E.)

$$\gamma(p) = (\sin(2^0\pi p), \cos(2^0\pi p), \ldots, \sin(2^{L-1}\pi p), \cos(2^{L-1}\pi p))$$

$$L = \begin{cases} 4, & \textit{with 2D input} \\ 10, & \textit{with 3D input} \end{cases}$$

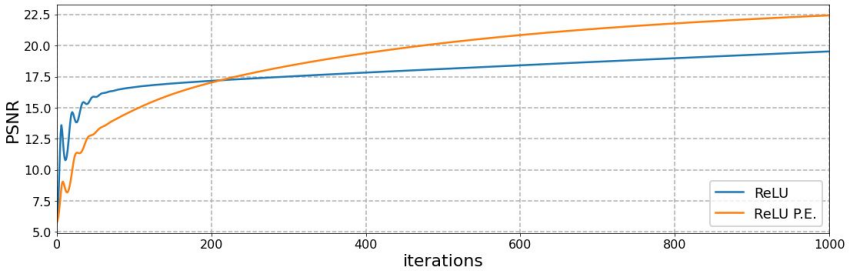
The encoding function maps the input coordinates into a higher dimensional space, before it is sent to the original network:

- Better representation of high-frequency image sections
- Faster convergence for high resolution representations

P.E. results on ReLU

Better performance and convergence speed for PSNR

High definition details



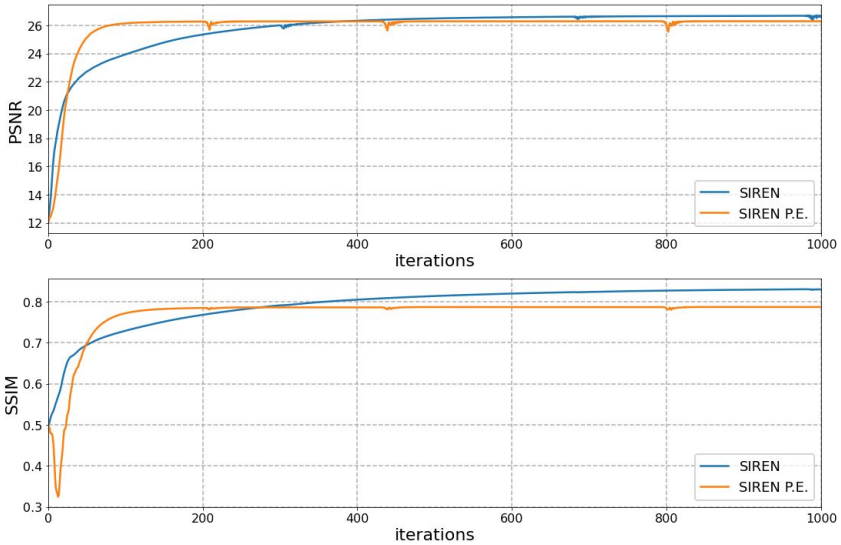
Skimage camera PSNR and SSIM

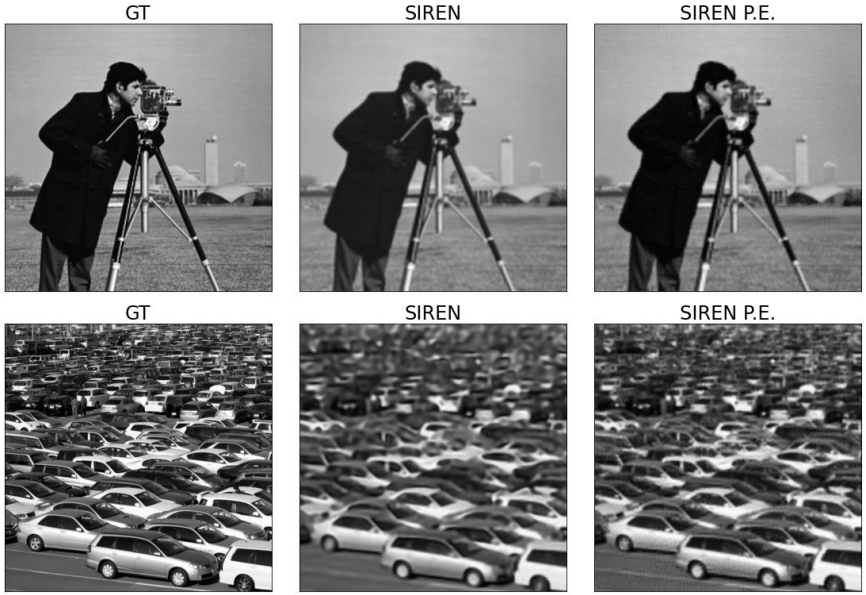
ReLU with 3 hidden layers, 512 hidden features each. Training on 1,000 steps, LR 1e-4

P.E. results on SIREN

Faster convergence

Slightly worse final results





Skimage camera PSNR and SSIM

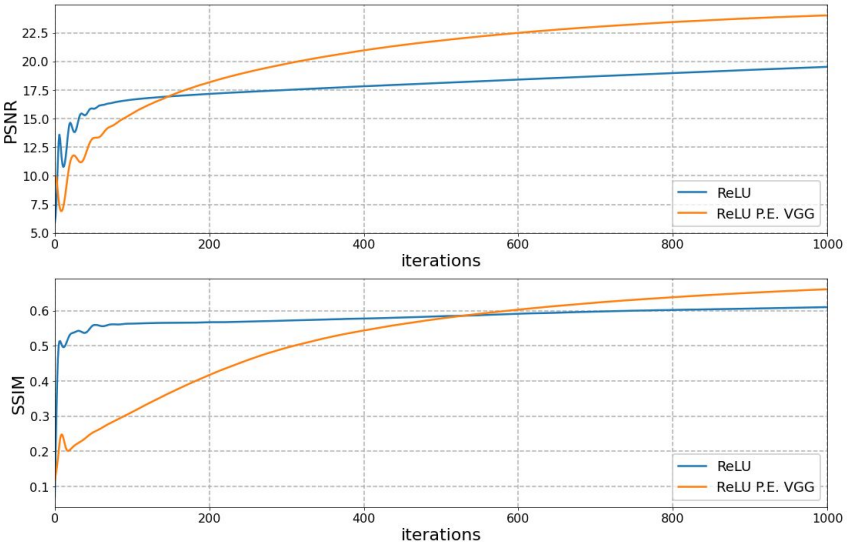
SIREN with 3 hidden layers, 512 hidden features each. Training on 1,000 steps, LR 1e-4

With mixed approach on ReLU

Best performance, high definition details

Lower convergence speed

Better than the single approach (VGG or P.E.)



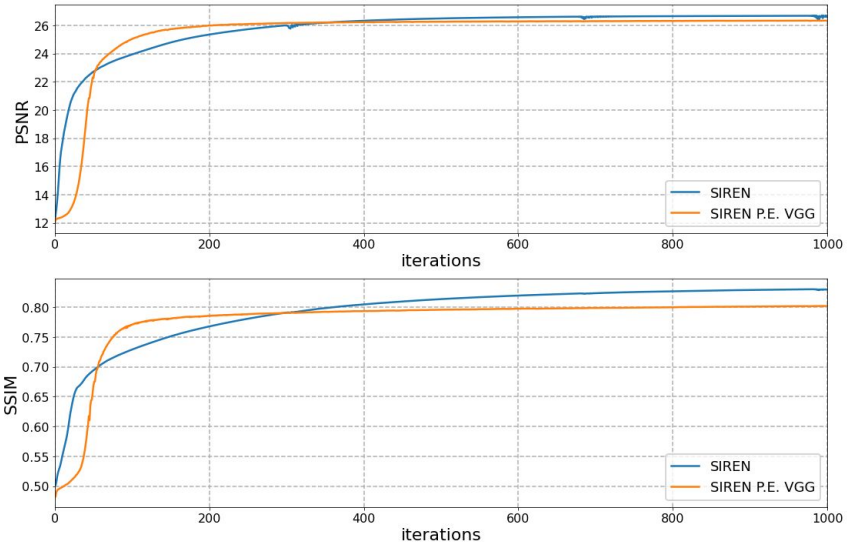
Skimage camera PSNR and SSIM

ReLU with 3 hidden layers, 512 hidden features each. Training on 1,000 steps, LR 1e-4

With mixed approach on SIREN

The mixed approach gets the worst results

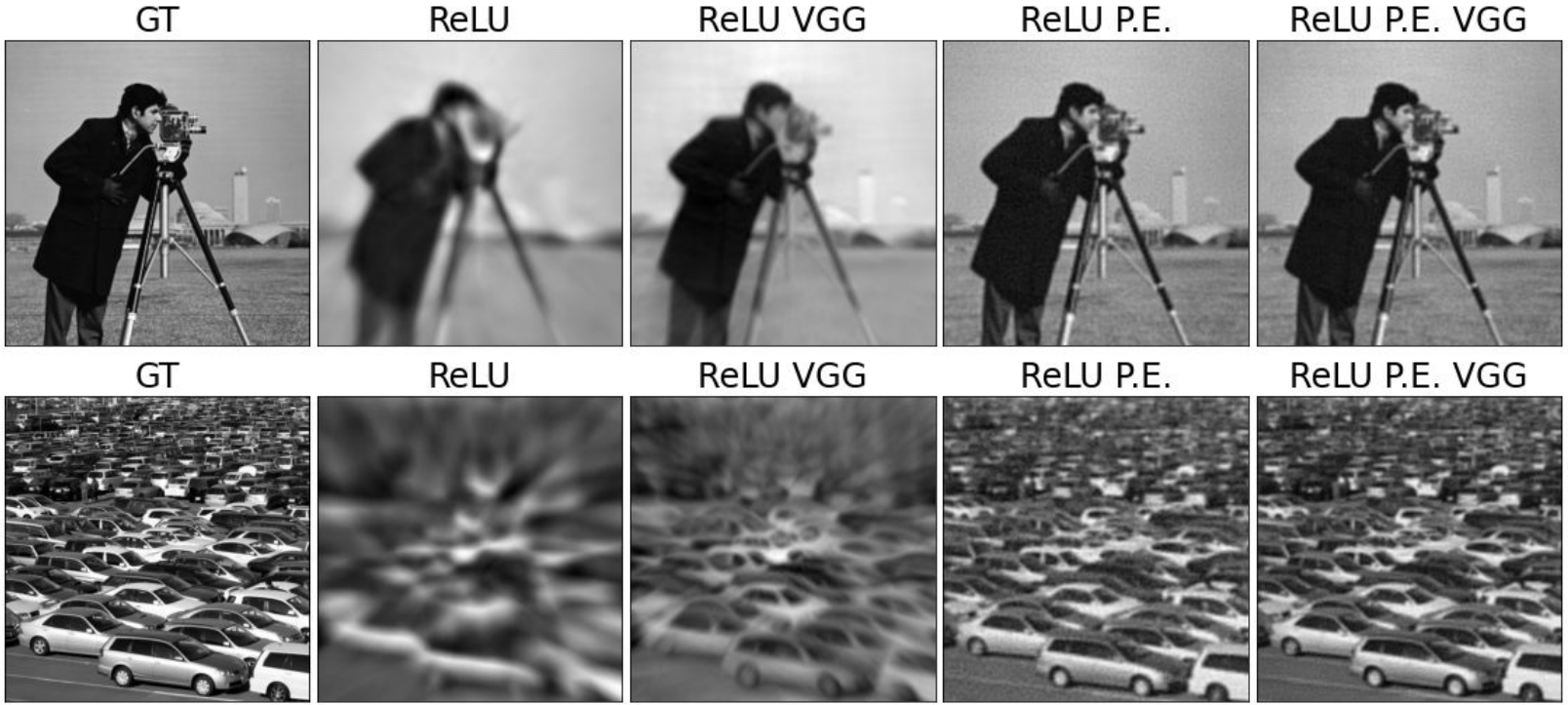
Low performance and convergence speed



Skimage camera PSNR and SSIM

SIREN with 3 hidden layers, 512 hidden features each. Training on 1,000 steps, LR 1e-4

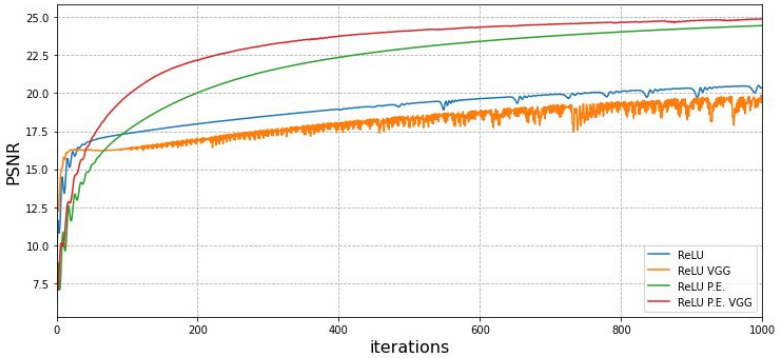
Compare all approaches on ReLU

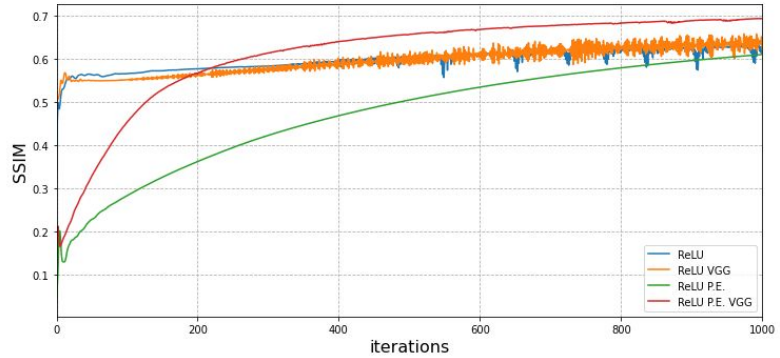


ReLU with 3 hidden layers, 512 hidden features each. Training on 1,000 steps, LR 1e-4

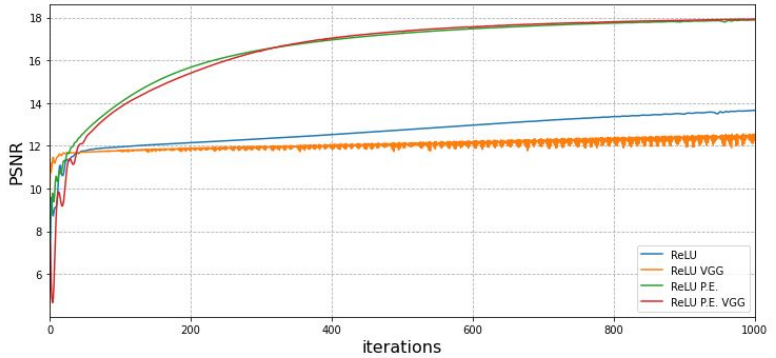
Compare all approaches on ReLU

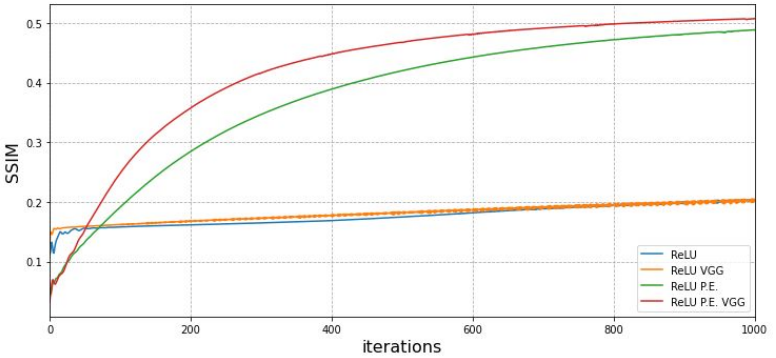
Skimage camera





Div2K 143



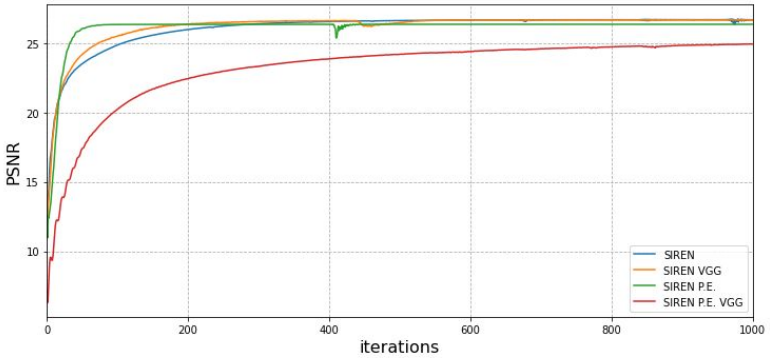


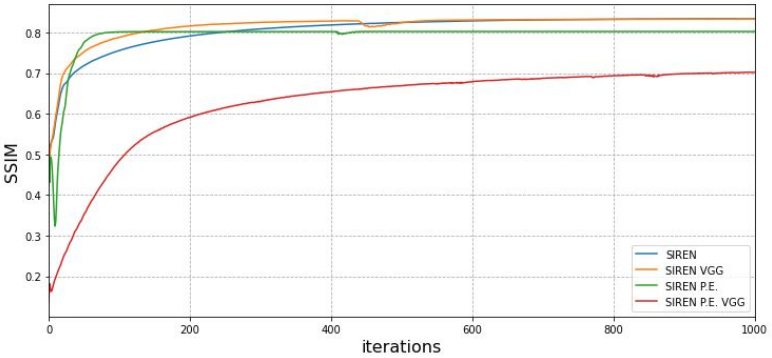
Compare all approaches on SIREN

SIREN with 3 hidden layers, 512 hidden features each. Training on 1,000 steps, LR 1e-4

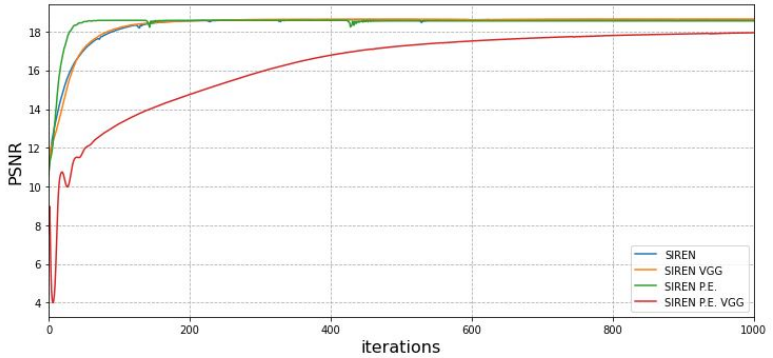
Compare all approaches on SIREN

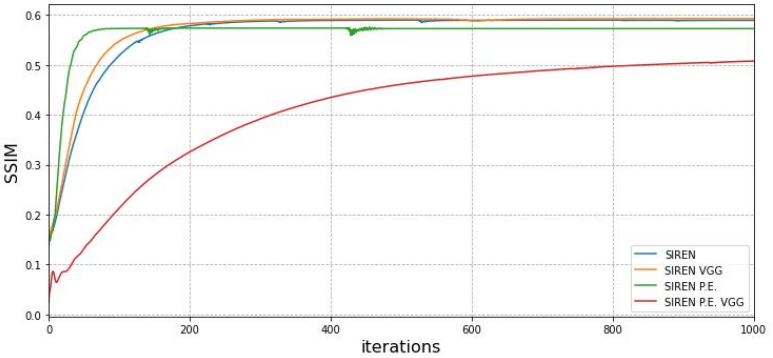
Skimage camera





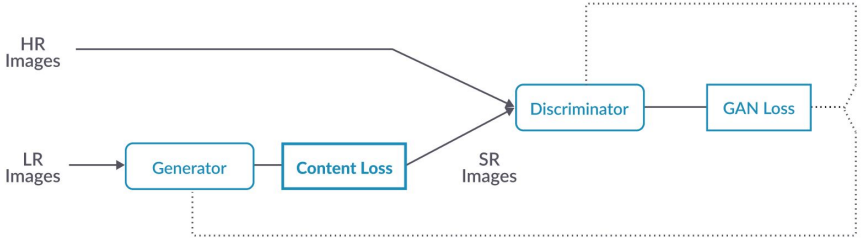
Div2K 143





Model comparisons

SRGAN



Super Resolution Generative Adversarial Network: state-of-the-art for SISR

- *VGG perceptual loss,* that is the weighted sum of:
  - Feature-based content loss
    - VGG loss, instead of the classic MSE one
  - Adversarial Loss
    - Encourages the network to favor solutions that reside on the manifold of natural images, by trying to fool the discriminator network
- Able to recover photo-realistic textures from heavily downsampled images

BICUBIC interpolation

- *Interpolation-based* method
- Convolution with a cubic kernel
- Standard baseline for comparison on SISR tasks, since 1981
- Good performance for small-medium scaling (X2, X4), not for higher scales (X8)

Model comparisons

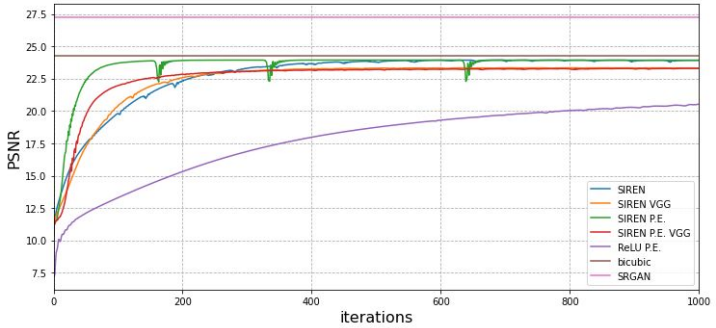
Compare ReLU, SIREN (with many flavours) and SRGAN networks between each other

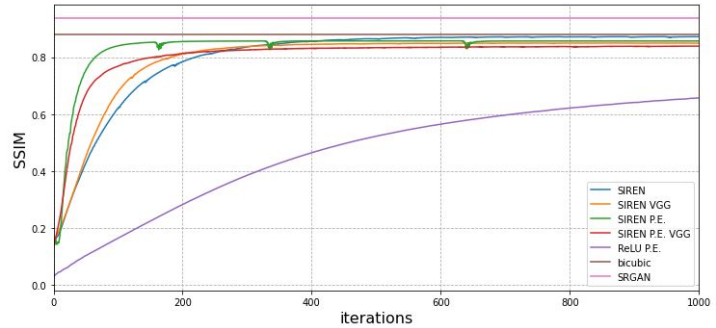
Reference with ground truth and bicubic interpolation function

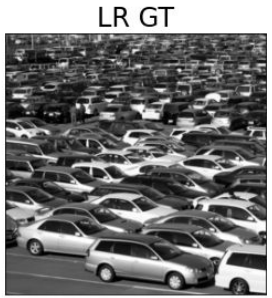
Different scaling factors:

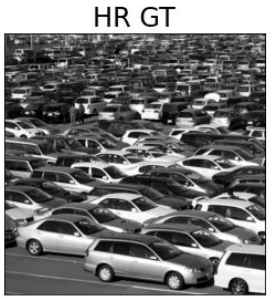
- X2
- X4
- X8

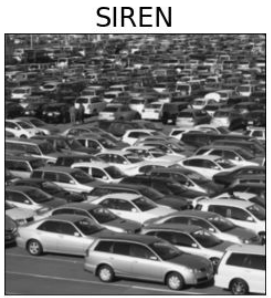
X2 scale (256 to 512)

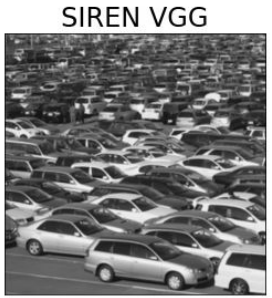


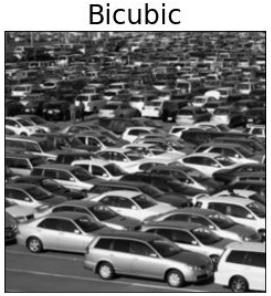


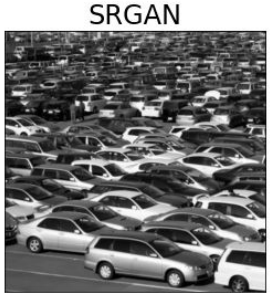






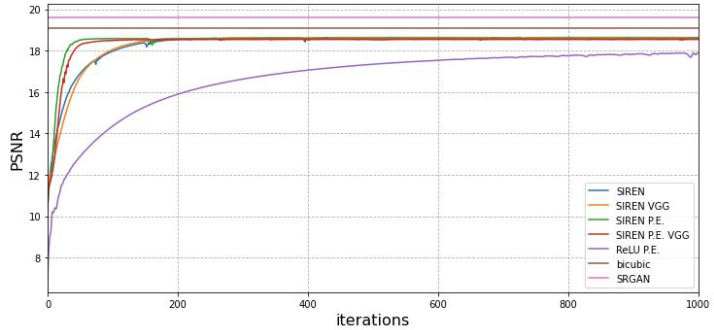


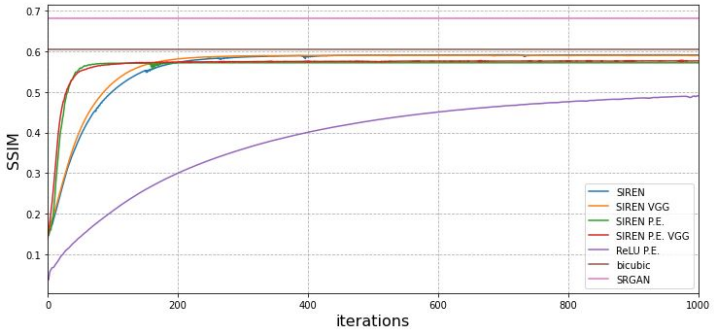


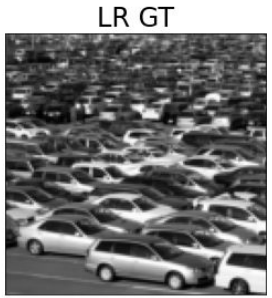


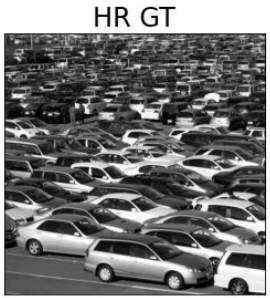
ReLU and SIREN with 3 hidden layers, 512 hidden features each. Training on 1,000 steps, LR 1e-4

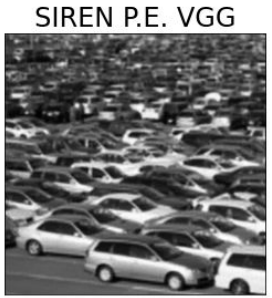
X4 scale (128 to 512)

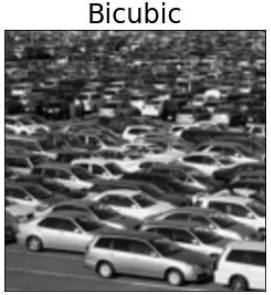


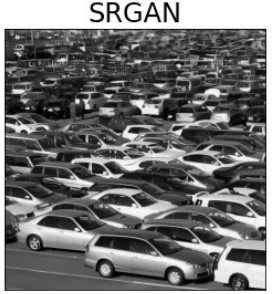






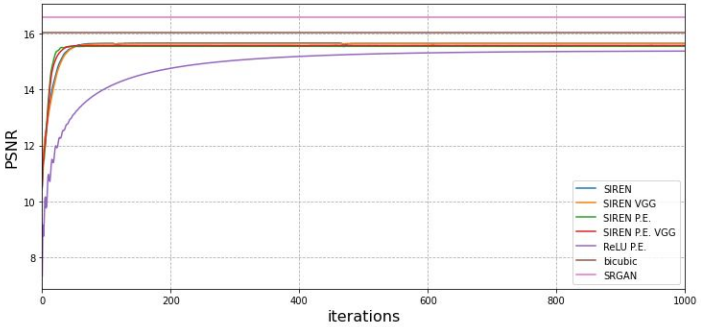


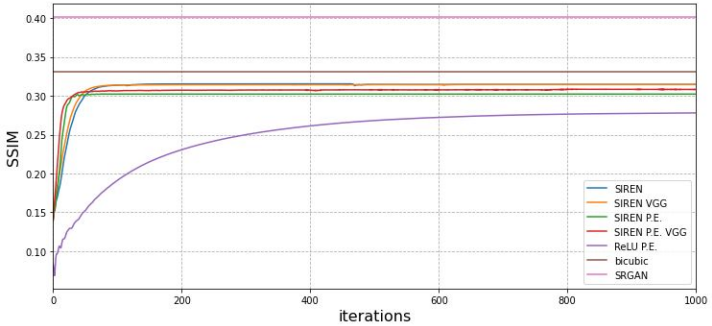


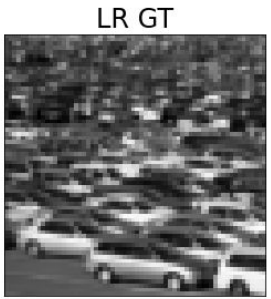


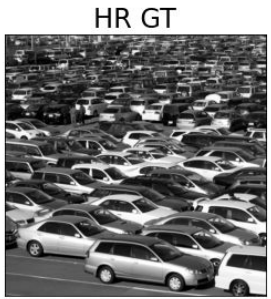
ReLU and SIREN with 3 hidden layers, 512 hidden features each. Training on 500 steps, LR 1e-4

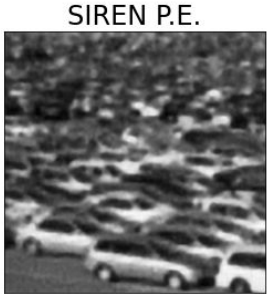
X8 scale (64 to 512)

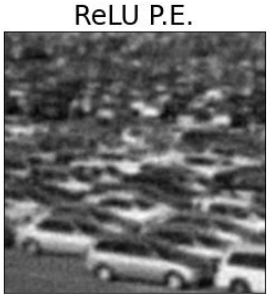


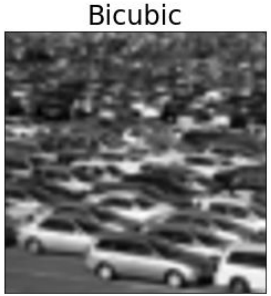


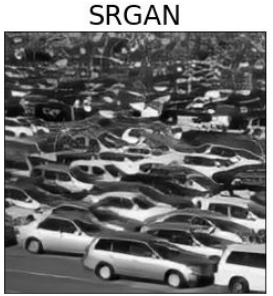












ReLU and SIREN with 3 hidden layers, 512 hidden features each. Training on 1,000 steps, LR 1e-4

CONCLUSIONS

Conclusions

Implicit neural representation

Pros:
- Great potential for signal representation and manipulation
- Faster convergence if employed with SIRENs
- Easy to implement

Cons:
- Cannot recover high frequency contents of the signal when upsampling

SIREN

Pros:
- Flexible in its applications
- Enables signal reconstruction from its derivatives

Cons:
- Needs attention for parameters initialization
- Not sensitive to the high-frequency details reconstruction solutions proposed

THANK YOU