Efficient Parameter Editing in Implicit Neural Representations for Image Processing Tasks

Emre Demirci
Artificial Intelligence and Data Engineering
Istanbul Technical University
150220317
demirciem22@itu.edu.tr
github:

***Abstract*—Conventional data representations discretize signals, leading to inneficient storing and constrained resolutions. Implicit Neural Representations (INRs), presents a different approach, modeling the signals as continuous functions using neural networks. This enables high quality reconstruction while using the memory efficiently and allows direct parameter manipulation for some image processing tasks such as denoising, deblurring and inpainting. This research investigates effective training methods for INRs, explores architecturals designs that enable efficient parameter editing without explicit decoding and identifies beneficial inductive biases for INR modification.**

I. Introduction

The representation of visual data has evolved, from storing pixel values to efficient neural architectures. Conventional data representations discretize signals, constraining resolutions and much more.

Implicit Neural Representations (INRs) approaches signal representation from a different viewpoint [1]. Unlike conventional discrete representations, INRs model signals as continuous functions utilizing neural networks, offering storing high resolution efficienctly, also it enbales theoretically infinite resolution, making them desirable for high quality reconstruction tasks.

Another advantage of INRs is that they allow direct parameter editing to achieve different image processing tasks such as denoising, deblurring and inpainting without requiring conversion back to original discrete image domain [1].

II. Problem Statement and Research Questions

Despite the capabilities of INRs, there are several challenges. The training of INRs for effective downstream task performance requires careful consideration of optimization strategies, loss functions, and network architectures.

This research proposal addresses three critical research questions:

A. Research Question A

How can INRs be trained effectively to facilitate downstream image processing tasks?

The training of INRs depends on factors such as the activation functions, number of neurons at hidden layers and number of hidden layers. To answer the question *"How can INRs be trained effectively to facilitate downstream image processing tasks?"*, we conducted experiments with various parameters.

B. Research Question B

What architectural designs allow for efficient parameter editing in INRs without explicit decoding or sampling?

The architectural design of INRs significantly impacts their amenability to parameter editing. Identifying design principles that enable direct parameter manipulation while maintaining computational efficiency represents a crucial research challenge.

C. Research Question C

What inductive biases are beneficial for training models that enable seamless INR modification?

Appropriate inductive biases during training can enhance the performance of parameter editing operations. Knowing which biases enable seamless modification is essential for developing INR-based image processing systems.

III. Related Work

Implicit Neural Representations has gained attention in recent years. Researches demonstrated the capability of neural networks to represent continuous signals with high fidelity [1].

Recent advances in INR-based image processing have shown promising results in tasks such as deblurring, denoising, and inpainting [4].

IV. Experiments

Troughout the experiments, MSE being less than 1e-5 is used as early stopping condition. Also all experiments are done using 256x256 grayscale Cameraman image showed in Fig. 1.

A. Research Question A

To analyze how to train INRs effectively, we conducted experiments on 3 different parameters: Activation Functions, Number of Neurons at Hidden Layers and Number of Hidden Layers.

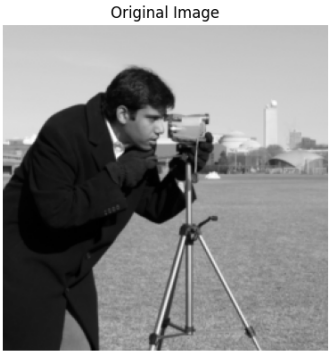

Fig. 1. Original 256x256 Cameraman image

*1) **Experiment 1: Activation Functions**:* In order to find how to train INRs effectively, 3 different activation functions were used seperetaly: sine [1], ReLU [2] and tanh [3]. Table I shows the results for different activation functions. We found that sine activation function is able to represent the data much better than ReLU and tanh as supported by the following test results. Figures 2, 3 and 4 show the reconstructed images and differences between reconstructed and original image using the stated activation functions. Loss curves and more details can be found on the shared colab link.

TABLE I
COMPARISON OF DIFFERENT ACTIVATION FUNCTIONS

Activation	Epochs	PSNR	MSE	Time
sine	3514	56.12dB	0.00001	97s
ReLU	10000	26.08dB	0.00986	205s
tanh	10000	24.04dB	0.015785	208s

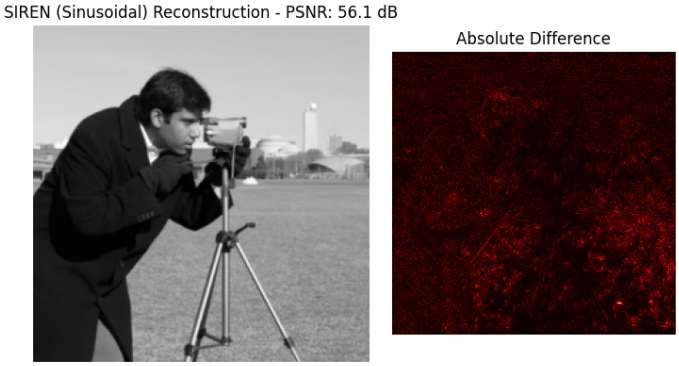

Fig. 2. Reconstructed Cameraman image by sine

As a result of this experiment, sine activation function is used for the rest of the experiments.

*2) **Experiment 2: Number of Neurons at Hidden Layers**:* We tried 3 different number of neurons at hidden layers: 128, 256 and 512. Table II shows the comparision for different values of number of neurons at the hidden layers. Figures 5, 6 and 7 shows the reconstructed images. Loss curves and more details can be found on the shared colab link.

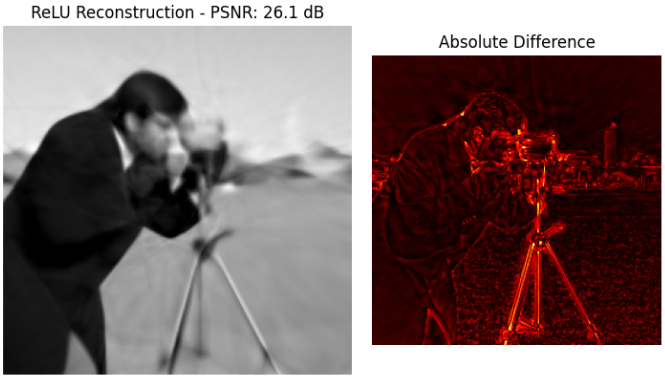

Fig. 3. Reconstructed Cameraman image by ReLU

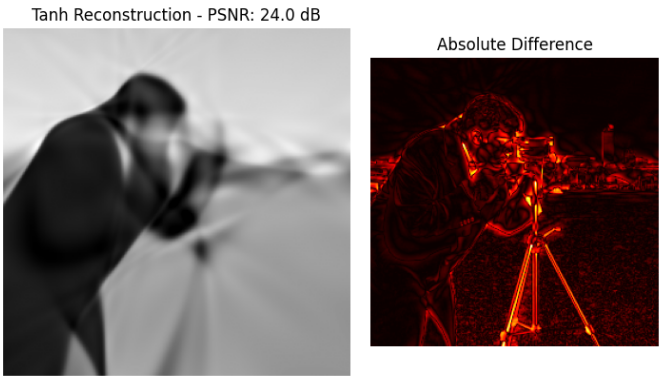

Fig. 4. Reconstructed Cameraman image by tanh

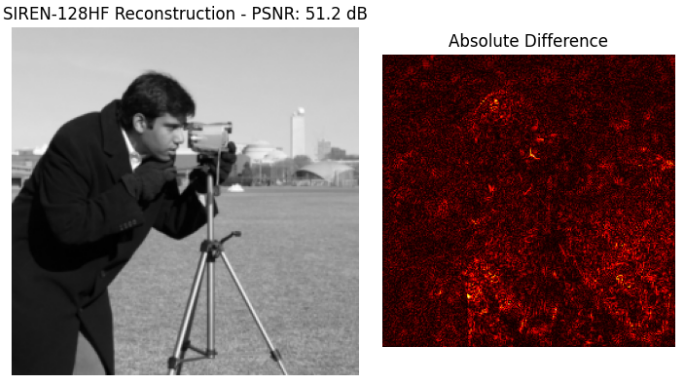

Fig. 5. Reconstructed Cameraman image by 128 neurons at hidden layers

For the rest of the experiments, we used 256 neurons for hidden layers because it performs as good as 512 with less parameters, making it desirable considering storage usage.

*3) **Experiment 3: Number of Hidden Layers**:* We tried 3 different number of hidden layers: 1, 2, 3 and 4. The comparision can be found on the Table III. Figures 8, 9, 10 and 11 shows the reconstructed images. Loss curves and more details can be found on the shared colab link.

For the rest of the experiments, we used 3 hidden layers because it performs nearly same as 4 hidden layers but uses

TABLE II
PERFORMANCE COMPARISON OF NEURAL NETWORK CONFIGURATIONS

#Neurons	Epochs	PSNR	MSE	Time
128	10000	51.20dB	0.00003	115s
256	3806	56.11dB	0.00001	105s
512	1693	56.20dB	0.00001	111s

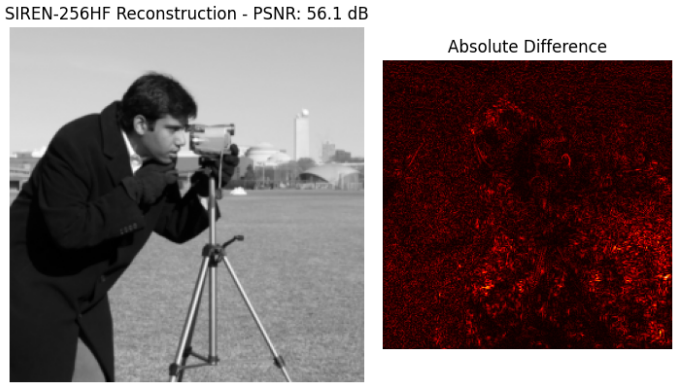

Fig. 6. Reconstructed Cameraman image by 256 neurons at hidden layers

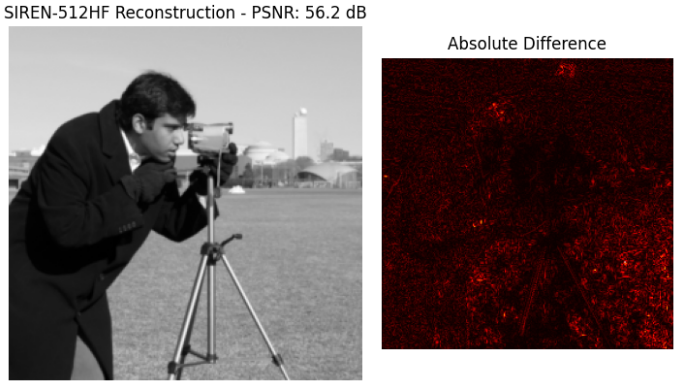

Fig. 7. Reconstructed Cameraman image by 512 neurons at hidden layers

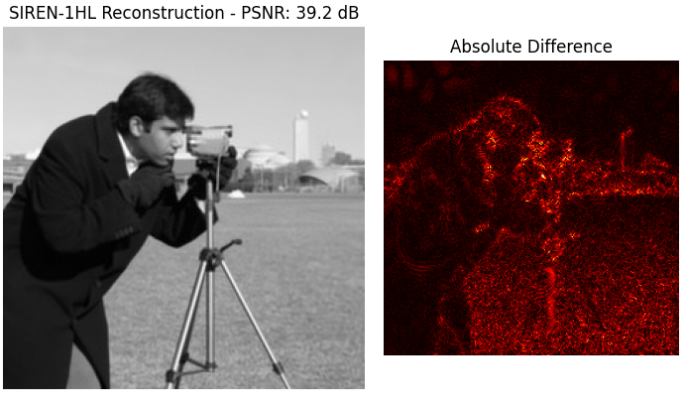

Fig. 8. Reconstructed Cameraman image by 1 hidden layers

less parameters and converges much faster than 2 hidden layers.

TABLE III
PERFORMANCE COMPARISON OF NEURAL NETWORK CONFIGURATIONS

#HiddenLayers	Epochs	PSNR	MSE	Time
1	10000	39.20dB	0.00004	148s
2	10000	54.0dB	0.00002	212s
3	3967	56.00dB	0.00001	110s
4	2783	56.10dB	0.00001	95s

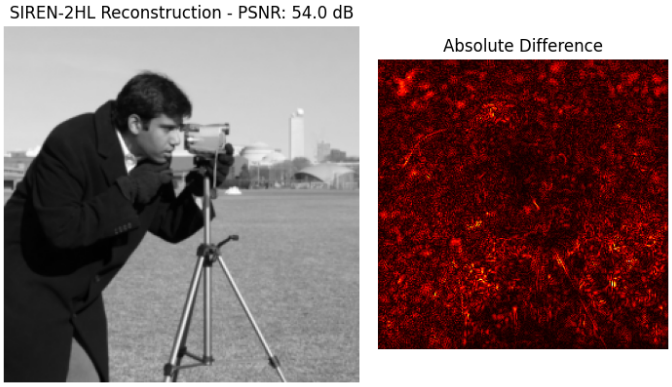

Fig. 9. Reconstructed Cameraman image by 2 hidden layers

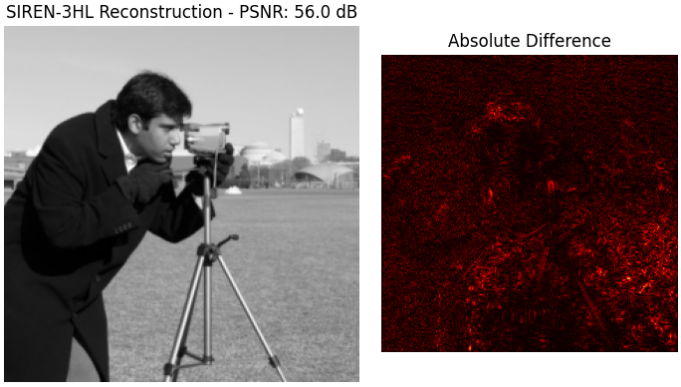

Fig. 10. Reconstructed Cameraman image by 3 hidden layers

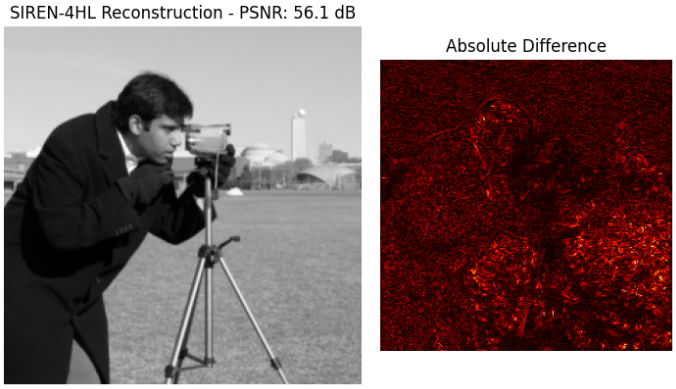

Fig. 11. Reconstructed Cameraman image by 4 hidden layers

B. Research Question B

I will answer this question based on findings of [4]: architectures that allow taking multiple derivatives allows efficient

parameter editing without explicit decoding and sampling because we can approximate the convolution operation by a linear combination of gradients, for more details and formal proof, please refer to [4].

I would love to conduct experiments for this question, however when I tried to take multiple derivatives of my models' parameters, I got out of memory errors which I could not solve , so I tried to approximate the results using only the first 10 derivatives, which makes 21 features compared to 69 features in the original INSP paper. Additionally, since my computational resources are much less than original INSP authors and they did not share their model's weights, I was not able to come even close to their results.

Nevertheless, here is my findings by using the original INSP-Net architecture trained on 1 example for 1000 epochs:

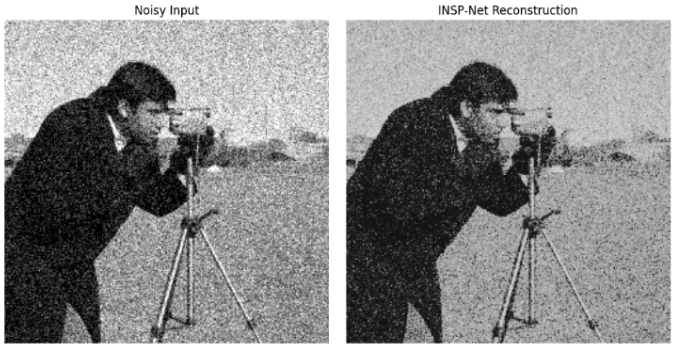

Fig. 12. Denoising

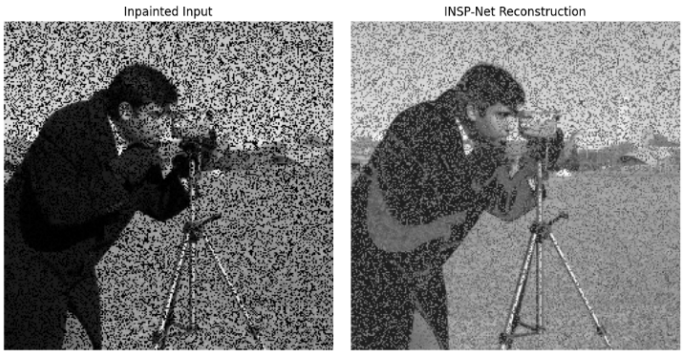

Fig. 13. Inpainting

If I had the computational resources, I would create this experiment as follows:

- Experiment 1) Create different INSP architectures, the original implementation uses the following architecture: input - 4x (FC - LeakyReLU) - FC, I would create different architectures and test them.
- Experiment 2) Original implementation uses 69 input features, which is equal to using first 34 derivatives, I would try to train same architecture using different number of derivatives and compare the results.

Fig. 14. Deblurring

- Experiment 3) Original implementation of INSP uses derivatives only respect to X, I would also like to conduct an experiment on that too.

C. Research Question C

Since I cannot use INSP network as I stated in the Research Question B section, I cannot conduct experiments in this part too. However, [4] explains that shift invariance and rotation invariance are beneficial inductive biases for training models that enable seamless INR modification. For more details and formal proof, please refer to [4].

And to create experiments for this research question, I would train 2 same model architectures with different training sets, for example train model A using base INRs and train model B using shifted/rotated INRs, then compare their performance on same test sets and shifted/rotated versions of that test sets and check if the results are similar.

V. Conclusion

This research tries to find answers to the questions regarding the design and training of Implicit Neural Representations for image processing tasks. By experimenting on different training strategies, architectural designs, and inductive biases, this research aims to advance the practical usage of INRs in real-world image processing scenarios.

References

[1] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, *Implicit neural representations with periodic activation functions*, Advances in neural information processing systems, vol. 33, pp. 7462-7473, 2020.

[2] V. Nair and G. E. Hinton, *Rectified linear units improve restricted boltzmann machines*, in Proceedings of the 27th international conference on machine learning (ICML-10), pp. 807-814, 2010.

[3] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, *Efficient backprop*, in Neural networks: Tricks of the trade, pp. 9-48, Springer, 2012.

[4] D. Xu, P. Wang, Y. Jiang, Z. Fan, and Z. Wang, *Signal processing for implicit neural representations*, in Advances in Neural Information Processing Systems (NeurIPS), 2022.