

## A APPENDIX

### A.1 EXPERIMENTAL SETUP

Our numerical implementation utilized the PyTorch framework and the Adam optimizer with a base learning rate of 0.001, which decays by 0.1 after each training epoch. All presented results were obtained using an MLP with three hidden layers, each containing 300 hidden nodes, and activation parameters drawn from a uniform distribution with support between 0 and 1. The training and testing processes were executed on a single 24GB NVIDIA RTX A6000 GPU. For performance evaluation, we used Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) for images, and Intersection over Union (IoU) for occupancy fields

### A.2 ADDITIONAL EXPERIMENTS

#### A.2.1 IMAGE SUPER RESOLUTION

As the learning process of an INR enables the discovery of a continuous functional mapping from low-dimensional spatial coordinates to high-dimensional signal space, one gains the ability to query the learned representation as needed. Consequently, unlike grid-based image representations, this representation becomes decoupled from the spatial resolution of the image, allowing for image super-resolution through interpolation of the learned representation.

For the purpose of demonstrating image super-resolution, we selected two images with different levels of detail complexity. The first, referred to as the "boy image" from the Set 14 Dataset (git), is a dynamic image that contains sharp frequency contents throughout the image, which could potentially challenge the detail retrieval capabilities of super-resolution. The second image, a woman, where sharp frequency details are only concentrated primarily around facial features, offering a localized test for the super-resolution process (See figure 8). Both images were downsampled by a factor of two and then used to train the INR. The trained INR, unconstrained by spatial resolution, was then used to infer high-resolution images.

The performance of image super-resolution demonstrates the generalizability of an INR. As evident from the results, *AINR* consistently exhibits not only the best PSNR and SSIM values across both cases but also the most visually coherent result, irrespective of the complexity of the image even with randomly initialized activation function parameters, when compared to existing INRs. Among the INRs, WIRE shows a notable deficiency in generalizing to a large set of unseen coordinates during testing, particularly evident in complex images. For example, in the case of the "boy image," WIRE struggles to generalize the down-scaled image's implicit mapping. When attempting super-resolution, WIRE introduces a significant number of random values for unseen coordinates, highlighting its limitations in handling images with varying complexity levels. Conversely, in the "women image," WIRE demonstrates a more reasonable representation. A similar situation can be seen from both GAUSS and MFN. Therefore, indicating a dependency of WIRE, GAUSS, and MFN's super-resolution capabilities on the detailedness of an image. These methods tend to excel with images where details are localized, such as the "women image," but fail with highly detailed images like the "boy image," revealing the unreliability of their learned implicit neural representations.

A closer examination of SIREN's super-resolution images reveals a learned low-passed representation, missing high-frequency components in the decoded high-resolution image. In contrast, *AINR* exhibits unparalleled performance in all scenarios tested, confirming the robustness and reliability of its learned representation. The performance of *AINR* is consistent across different types of images, establishing it as the only architecture capable of learning a reliable, continuous representation of images. These findings further corroborate our hypothesis: an INR equipped with an appropriately sequenced activation framework learns a superior representation than a fixed sequence of activations that depends on the given signal.

#### A.2.2 EDGE DETECTION

INRs are characterized by their inherent ability to encode signals in an implicit manner, where a key attribute of a generalized INR lies in its ability to undertake tasks which are typically reserved for

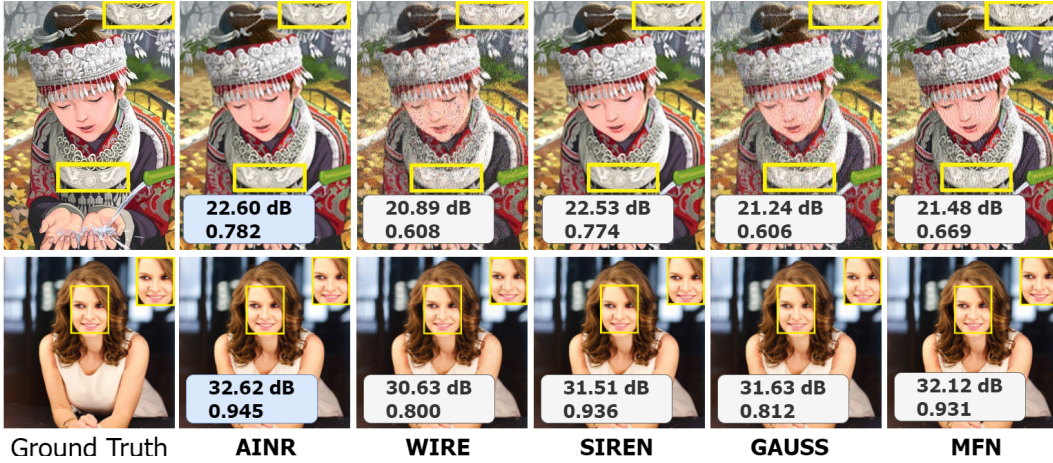


Figure 8: **Image super resolution capabilities of AINR.** The top row showcases super resolution results for the "boy image," where it contains a rapidly varying dynamic content throughout the image, while the bottom row depicts a woman's picture with localized frequencies around the face region. AINR emerges as the sole INR consistently delivering the highest PSNR and SSIM scores for both cases, showcasing its superiority in learned implicit neural representation.

explicit representations. With the explicit signal representations, especially with images, the spatial filters like Sobel, Gaussian, Canny, etc enable the extraction of edges. This process is essential for activities like identifying the boundaries of objects, detecting text, or recognizing facial features, all of which rely significantly on precise edge detection (Ziou et al., 1998). Even though it's possible to obtain edge maps from an INR through its decoded outputs, INRs provide the benefit of representations that are learned and differentiable. Therefore, an INR must not only precisely capture the signal but also facilitate the retrieval of its gradient data. For an INR to efficiently function as an edge detector, an INR requires to encode pixel-level relationships within its weights and biases.

The capabilities of extracting edge maps using INRs become evident when applied to an image which has clear visual edges, like the Monarch picture from the Set 5 Dataset (git). To identify the possibility of edge detection capabilities of INRs, initially, the image representation has been learned through training the INR. Thereafter, the gradient operator is utilized between this learned model and the coordinates used during training. Figure 9 illustrates the RGB image of the Monarch alongside its edge maps derived from INRs.

AINR stands out by providing the cleanest and most well-defined edge map, identifying necessary edges with minimal false edge identifications. Among other INRs, while MFN can identify edges, its edge map often contains excessive texture content, potentially leading to false edge detection. GAUSS on the other hand, produced smooth edge maps, which can diminish the true edge signal and result in a less reliable representation. In contrast, AINR preserves intricate patterns, offering a more accurate portrayal of the butterfly's natural markings. SIREN also demonstrates its effectiveness as an edge detector, leveraging its built-in capabilities for identifying edges. However, even in the absence of such inherent advantages, AINR excels in detecting edges, highlighting its proficiency in capturing complex patterns and delivering precise representations. Therefore, AINR showcases its effectiveness as a reliable edge detector compared to existing state-of-the-art INRs.

### A.3 HIGH FREQUENCY ENCODING CAPABILITIES

A straightforward method to evaluate the high-frequency encoding capabilities of INRs are by assessing their effectiveness in representing sharp frequencies. This can be achieved by examining a simple image comprised solely of sharp transitions, where the abrupt color changes in the spatial domain correspond to high-frequency components in the frequency domain.

For this experiment, we selected the image displayed on the left side of figure 10. INRs were then trained on this image. Once each INR had learned the implicit representation, it was decoded by in-

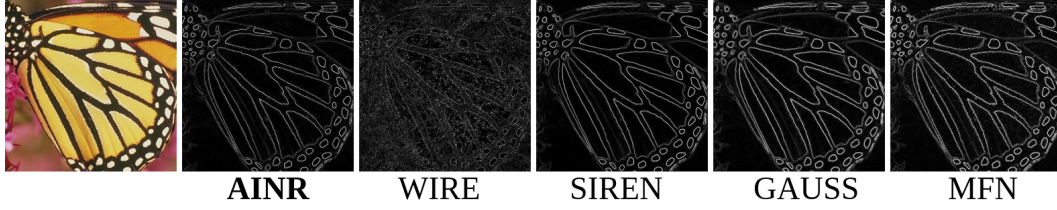


Figure 9: *AINR*'s edge detection capabilities: Edge maps obtained through applying the gradient operator on the learned representation.

putting the corresponding coordinates. Subsequently, we obtained normalized error plots comparing the ground truth with the decoded representations.

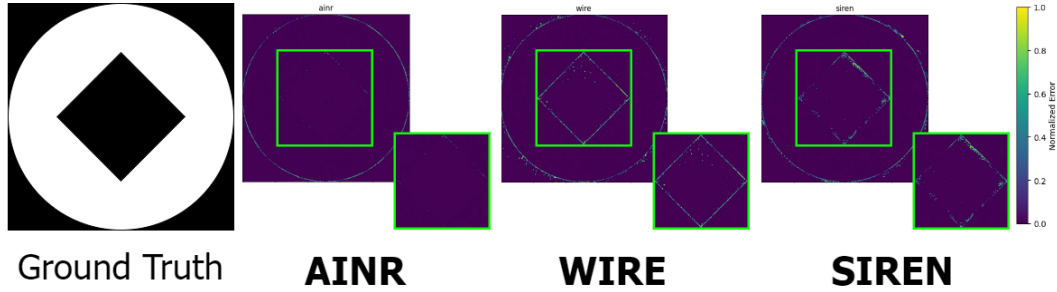


Figure 10: **Ability to encode high frequencies in INRs.** The results present normalized error plots obtained after training each INR on the ground truth image. It is evident that *AINR* exhibits the lowest error among the INRs when encoding signals into their weights and biases. Notably, it excels in encoding high frequencies exceptionally well compared to the other INRs as *AINR* is tailored to the given signal. Conversely, *WIRE* and *SIREN* demonstrate errors not only in encoding high-frequency content but also in constant color areas.

As illustrated in figure 10, *AINR* exhibits the smallest error margin between the decoded image and the ground truth, in comparison to other models such as *SIREN* and *WIRE*, which demonstrate noticeable errors even in areas of uniform color. This outcome distinctly highlights *AINR*'s superior capability in accurately encoding high-frequency components. Such precision in representation validates the underlying hypothesis of *AINR* in frequency domain perspective as well: the sequence of activations within an INR should adapt based on the characteristics of the input signal.

#### A.4 EXPLANATIONS

##### A.4.1 PSEUDO CODE OF AINR

For enhanced understanding of *AINR*, we provide its pseudocode in algorithm 1. It's important to highlight that *AINR* offers the flexibility to both expand and condense its dictionary. This means elements can be added to or removed from *AINR*'s dictionary as needed. For example, if we remove all activation atoms except for the Gabor Wavelet, *AINR* transforms into *WIRE* (Without *WIRE*'s pre-optimized parameters). This capability demonstrates that *AINR* act as the bridge between INRs, and providing a unified, and versatile framework capable of emulating existing INRs through minimal modifications to its dictionary.

##### A.4.2 SPATIAL DOMAIN VARIATION

In figure 11, the seven activation atoms used in *AINR* are depicted, showcasing their spatial domain variations. These activations include Sinc, Raised Cosine, Root Raised Cosine, Prolate Spheroidal Wave Function (PSWF), Gabor Wavelet, Sinusoid, and Gaussian. The diversity of these filters enables the *AINR* framework to capture a wide range of features, from sharp edges to smooth tran-

sitions, periodic patterns, and localized details. This variety ensures the network can represent different structures effectively, making the method highly adaptable.

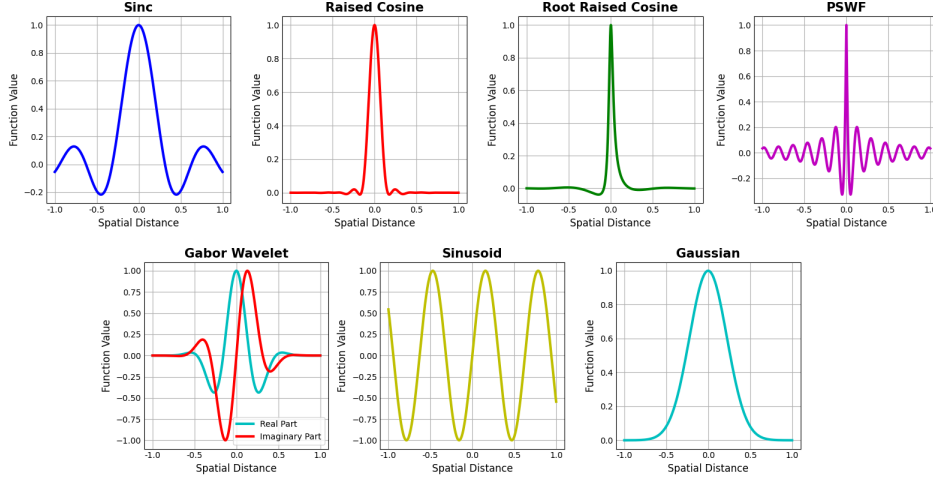


Figure 11: **Spatial domain activation function variation.** *AINR* benefits from all types of activations, as each has unique spatial and frequency characteristics.

---

#### Algorithm 1 :Pseudo Code of *AINR*

---

```

1: Input: Explicit signal representation
2: Output: Implicit Neural Representation
3: Initialization:
4:   Dictionary  $\leftarrow$  Dictionary of activation functions
5:    $N \leftarrow$  number of hidden layers
6:   Model  $\leftarrow$  MLP (single hidden layer, matching I/O dimensions)
7:    $\forall j \in \{1, \dots, N\}, \text{Model.activation}_j \leftarrow \text{None}$ 
8:   Minimum Loss $_j \leftarrow \infty$ , Best Activation $_j \leftarrow \text{None}$ ; for all  $j \in \{1, \dots, N\}$ 
9: for  $j \in \{1, \dots, N\}$  do
10:   for activation function in Dictionary do
11:     Model.activation $_j \leftarrow$  activation function
12:     for epoch  $\in \{1, \dots, \text{num\_epochs}\}$  do
13:       Train the Model
14:       Find Loss $_j$ 
15:       if Loss $_j <$  Minimum Loss $_j$  then
16:         Minimum Loss $_j \leftarrow$  Loss $_j$ 
17:         Best Activation $_j \leftarrow$  activation function
18:       end if
19:     end for
20:   end for
21:   Model.add a hidden layer
22:   Model.activation $_j \leftarrow$  Best Activation $_j$ 
23: end for

```

---

#### A.4.3 HOW DOES *AINR* DIFFER FROM BASELINES?

The *AINR*'s dictionary consists of seven activation functions, three of which, sinusoidal, Gaussian, and Gabor wavelet, have been previously introduced in other studies. This naturally raises the question of how *AINR* differs from these baselines. The key distinction lies in the initialization of activation function parameters. While earlier studies used these same activation functions, their parameters (such as  $\alpha$  and  $\beta$  in the sinusoidal function, as noted in section 3.2 of the main paper)



were typically fine-tuned via exhaustive grid searches for each specific INR application. In contrast, *AINR* initializes the parameters of every activation function randomly. As demonstrated in section 4.5 of the main paper, when baseline models (which use a single activation function throughout the network) are initialized with random activation parameters, they fail to achieve comparable performance. Therefore, unlike these baselines, *AINR* does not rely on any pre-optimized activation function parameters, offering a more flexible and robust approach.

#### A.4.4 TOTAL EPOCHS AND LEARNING CURVES

Compared to other INRs, *AINR*'s total number of epochs is determined by the size of the dictionary. Since *AINR* can accommodate any dictionary size, let us assume the dictionary contains  $k$  activations. For simplicity, consider a model with three hidden layers. If each activation is trained for  $x_1$ ,  $x_2$ , and  $x_3$  epochs in the first, second, and third layers respectively, the total number of epochs is given by  $k \times x_1 + k \times x_2 + k \times x_3$ . In our experiments, we set  $x_1 = 100$ ,  $x_2 = 100$ , and  $x_3 = 200$ . The training plot for the third image from the Kodak dataset is shown in figure 12. For the image representation task, we used seven different activation functions, resulting in a total of 2800 epochs. To ensure a fair comparison, all other baselines were trained for  $k \times (x_1 + x_2 + x_3)$  epochs. The convergence plot for the baselines is displayed in figure 13. Although *AINR* requires total 2800 epochs when using seven activations in the dictionary, it achieves approximately 40 dB by the 1600th epoch (see figure 12). In comparison, the baselines reach a maximum of around 37 dB, and even that is only achieved after approximately 2000 epochs.

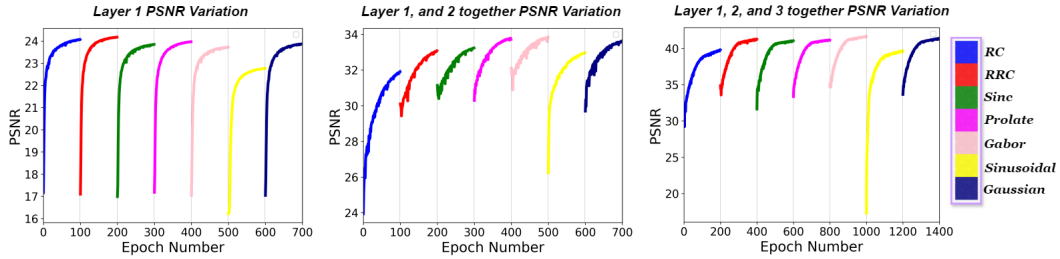


Figure 12: **PSNR variation of *AINR* with epochs:** *AINR*'s training procedure is sequential. Initially, it optimizes a single hidden-layer INR, changing the activation function of the first layer every 100 epochs. Once the matched activation for the first hidden layer is found, a second hidden layer is added. The network is then trained while changing the activation function of the second layer every 100 epochs. After determining the matched activation for the second layer, a third hidden layer is added, and the network is retrained, changing the activation function of the third hidden layer every 200 epochs.

### A.5 ADDITIONAL RESULTS

#### A.5.1 IMAGE REPRESENTATION

Unlike traditional INRs, which use the same activation function throughout the network with sub-optimal parameters that may not be ideal for the given signal, *AINR* offers a framework that allows the network to adapt its internal configuration to better match the signal. This added flexibility enables *AINR* to consistently outperform existing INR methods.

This is evident from our thorough evaluation of *AINR* on the Kodak image dataset, as shown in figure 3. In addition to the radar plot, we have also provided decoded representations of the Kodak dataset, which are presented in figure 14. Moreover, the average PSNR and SSIM metrics for *AINR* and baseline models across the dataset are summarized in table 2.

#### A.5.2 IMAGE INPAINTING

A key difference between traditional signal representation mechanisms and INRs is that INRs attempt to establish a continuous implicit functional relationship between normalized coordinates and signal values. The generalizability of this relationship depends heavily on the type of activation

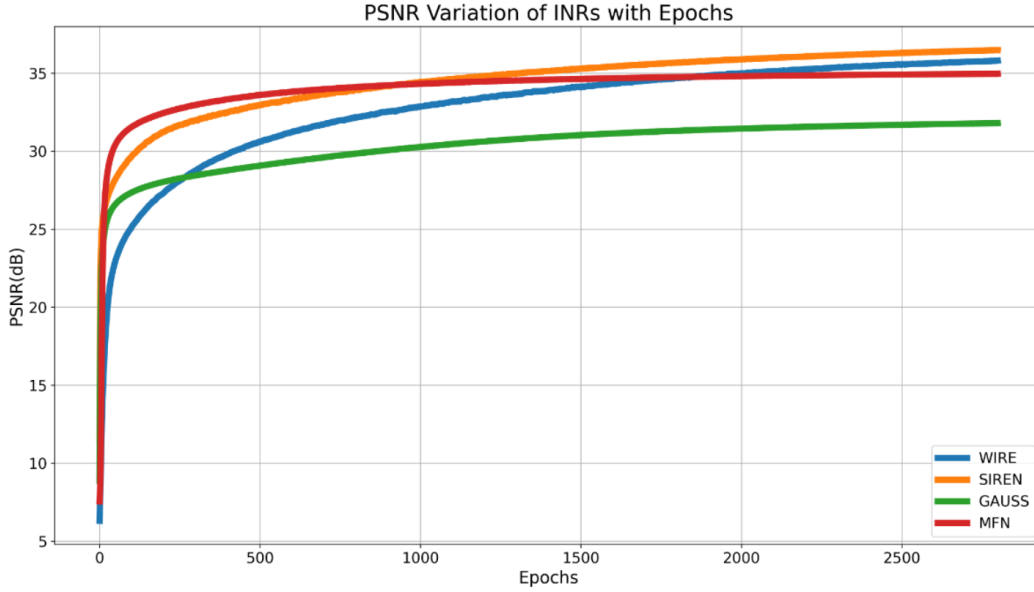


Figure 13: PSNR variation of the baselines for the third Kodak image



Figure 14: Additional image representation results from the Kodak Dataset: As shown by the results, AINR is the architecture that consistently delivers the cleanest image representation. Moreover, it not only excels in producing clean output but also achieves the highest accuracy metrics.

Table 2: Average metrics for image representation task across the entire Kodak dataset

Method	PSNR (dB)	SSIM
AINR	36.55	0.93
WIRE	32.30	0.90
SIREN	32.19	0.88
GAUSS	31.34	0.84
MFN	30.15	0.85

function used. AINR, by exploring its dictionary to find activation atoms that best match the given signal and task, consistently outperforms all INR baselines, regardless of the specific task.

This is clearly demonstrated by the thorough evaluation we conducted on the image inpainting task using the entire Kodak dataset. The PSNR variations across all methods are shown in figure 4. As shown, AINR consistently achieves the highest accuracy metrics for every image in the dataset. Sample inpainting outcomes are displayed in figure 15. The first column represents the ground truth

image, while the second column shows the text-masked image. The text mask used contains different fonts and sizes, with overlapping parts, and this same mask was applied to the entire dataset. This presents a significant challenge for INRs to recover the original signal. However, as seen, AINR recovers the original image with the highest accuracy. Unlike existing baselines such as WIRE, AINR does not overfit the training data, nor does it tend to produce low-pass signal representations lacking fast-varying components. AINR consistently delivers the highest accuracy metrics along with the most visually coherent inpainting outcomes, as it is specifically tailored to the given signal and task. The average PSNR across the dataset is presented in table 3. .

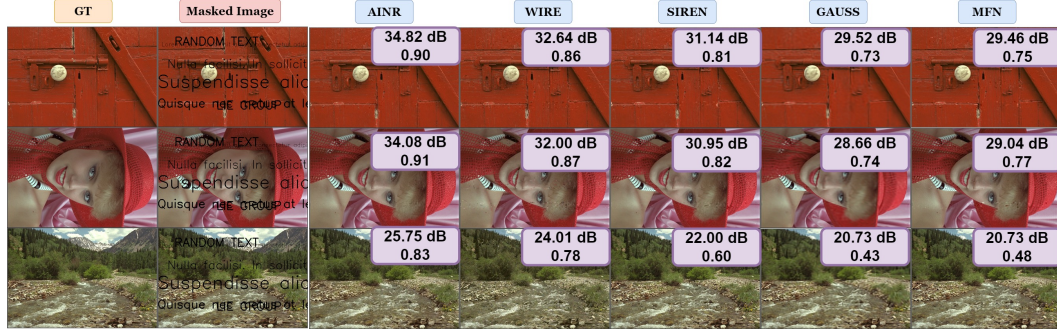


Figure 15: **Additional image inpainting outcomes for the Kodak dataset:** As demonstrated, AINR is the only INR that consistently delivers the highest accuracy metrics along with the cleanest image inpainting results, regardless of the image type. This can be attributed to its ability to dynamically tailor the sequence of activations to the specific signal and task, rather than relying on pre-optimized activation function parameters for every signal.

Table 3: Average metrics for image inpainting task across the entire Kodak dataset

INR	PSNR (dB)	SSIM
AINR	32.34	0.90
WIRE	29.82	0.85
SIREN	28.44	0.79
GAUSS	26.41	0.69
MFN	26.02	0.70

An additional image inpainting experiment has been conducted with a different text mask, and the results obtained are shown in figure 16. In this experiment, a randomly generated text mask with varying font sizes is used. As evident from the results, AINR is the only architecture that achieves the highest PSNR and SSIM values, indicating the best recovery compared to existing INRs.

### A.5.3 OCCUPANCY FIELDS

The complete decoded occupancy fields, along with the ground truth, are shown in figure 17. *AINR* consistently delivers the occupancy field that is closest to the ground truth. This capability is attributed to the ability of *AINR* to find the optimal activation sequence for each given signal.

### A.5.4 NEURAL RADIANCE FIELDS

In addition to the novel views provided in section 4.4, novel views from different viewing angles and positions are provided in the following figures for Chair and Hotdot datasets respectively,

## A.6 ABLATION STUDIES

### A.6.1 EFFECT OF NETWORK HYPERPARAMETERS

A study has been conducted to investigate the impact of varying the number of hidden nodes and layers, and learning rate on performance. The results related to changes in the number of hidden



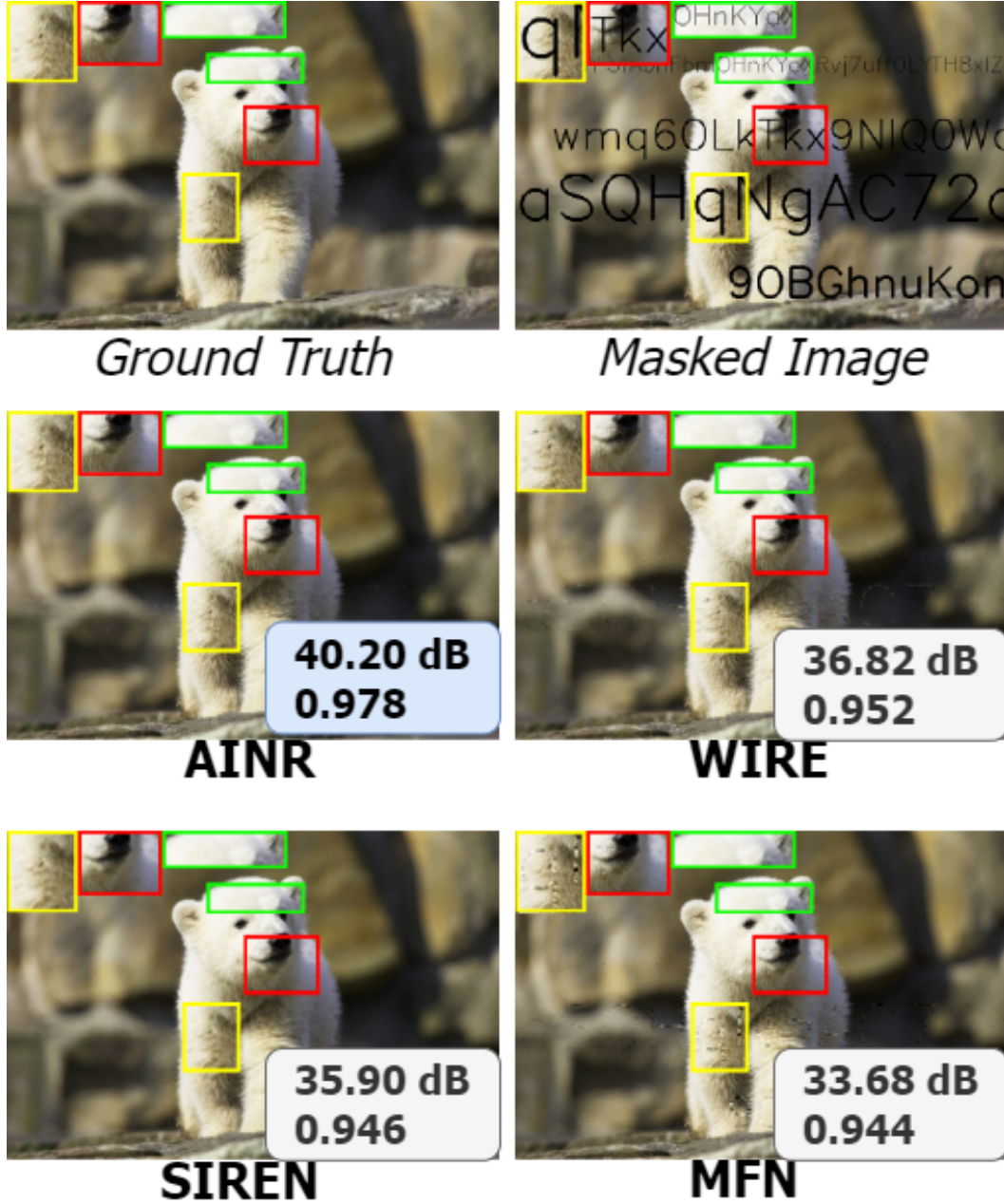


Figure 16: **Image inpainting capabilities of AINR.** The top row presents the ground truth and the masked image, where the text is random with varying font sizes. It is evident that AINR achieves the highest PSNR and SSIM values while producing the most visually coherent inpainting outcome among the considered INRs. This clearly demonstrates AINR’s superior generalization ability compared to current INRs. Conversely, other architectures such as WIRE and MFN exhibit signs of overfitting with the provided partial data.

nodes are shown in left plot of figure 20. It is evident that AINR surpasses all existing INRs regardless of the number of hidden neurons employed. This superiority is attributed to AINR’s ability to self-optimize based on the input signal, ensuring that the optimal sequence of activations yields superior results compared to pre-optimized INRs.

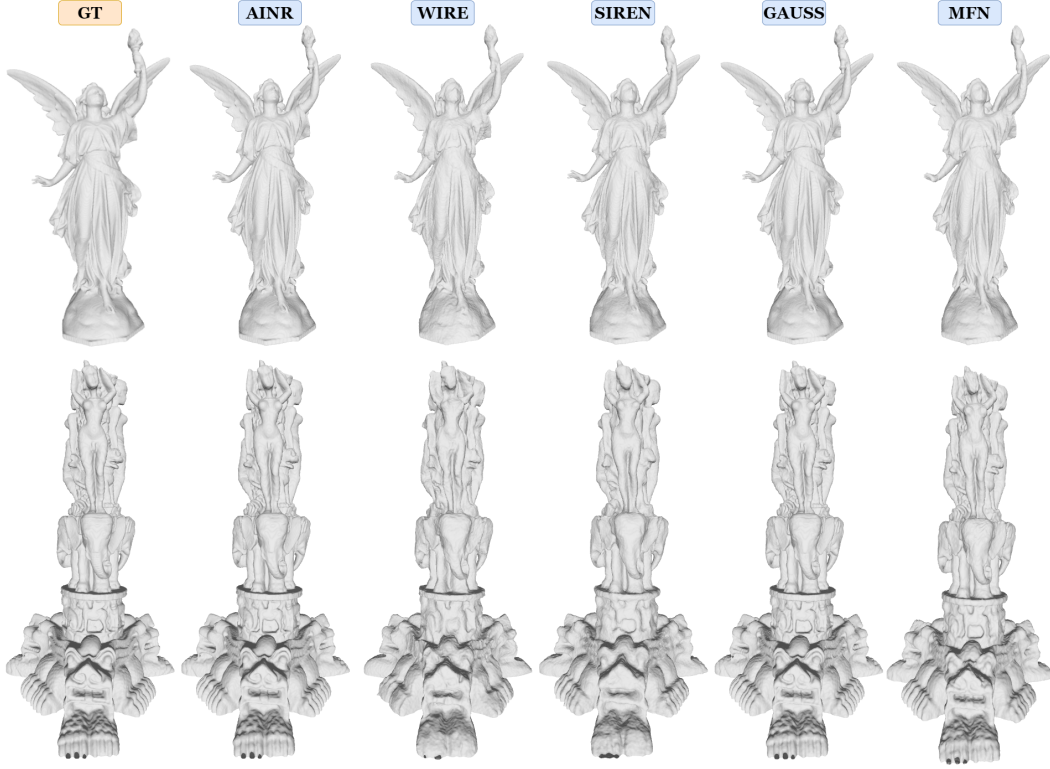


Figure 17: **Complete decoded occupancy fields:** It is evident from these representations that most baselines showcase a low-pass decoded representation, primarily due to the inability of their activation functions to effectively capture the rapidly varying components across the INR. In contrast, *AINR* delivers the occupancy field closest to the ground truth, achieving the highest IoU metric.

The middle plot of figure 20 depicts the effect of the number of layers on the PSNR. As can be seen, *AINR* demonstrates exceptionally competitive PSNR metrics with merely one or two hidden layers. This performance stands out in contrast to other INRs, which begin with activation parameters already tailored for convergence. However, *AINR* starts with random parameter initializations, highlighting its efficiency in self-optimization to reduce the loss between explicit and implicit representations. When the model configuration expands to include three hidden layers, *AINR*'s ability to adapt and tailor its approach for the specific signal allows it to outperform all existing INRs, showcasing its robust optimization capabilities.

Lastly, the right side plot of figure 20 illustrates *AINR*'s performance with log scale learning rate. As can be seen, *AINR* shows strong performance, maintaining a high PSNR around 40 dB at a low learning rate of  $10^{-3}$ . Even when the learning rate is increased by an order of magnitude to  $10^{-2}$ , *AINR* manages to sustain a relatively high PSNR. However, as the learning rate further increases to  $10^{-1}$ , there is a noticeable decrease in PSNR, although it still performs better than every baseline except MFN. On the other hand, WIRE shows significant sensitivity to the learning rate. It reaches its peak PSNR at a learning rate of  $10^{-2}$ , but its performance drastically drops as the learning rate increases further. This demonstrates that WIRE's optimal learning rate range is narrower, and its performance quickly deteriorates outside this range. SIREN also demonstrates a decline in performance, but its behavior is steadier compared to WIRE. As the learning rate increases, SIREN's PSNR decreases consistently, never reaching the peak values observed with *AINR* or even WIRE. It can be concluded that, *AINR* stands out for its ability to maintain high PSNR values across a wider range of learning rates, highlighting its robustness.





Figure 18: *AINR*'s novel view synthesis capabilities on the Chair dataset: A closer inspection of the novel views generated by *AINR* reveals that its synthesized images capture finer details, such as the intricate carvings on the top of the chair and its legs, to a much greater extent than the baselines. Additionally, *AINR* successfully preserves the gold pattern on the chair's cushion without excessively smoothing it, unlike the baselines, which tend to produce low-pass representations. This ability to retain both fine details and texture while avoiding over-smoothing highlights *AINR*'s superior performance in generating realistic novel views.



Figure 19: *AINR*'s novel view synthesis capabilities on the Hotdog dataset: Upon closer inspection of the novel views generated by *AINR*, it is evident that *AINR* produces more realistic and detailed representations compared to the baselines. Specifically, *AINR* captures the fine texture of the hotdog bun and the subtle lighting effects on the plate and condiments, which are much closer to the ground truth. In contrast, the baseline methods, such as WIRE and SIREN, tend to over-smooth the details, losing the sharpness in the condiments and the realistic shadows around the plate. *AINR* not only preserves the structure and texture of the hotdog but also handles the complex lighting conditions more effectively, resulting in the most visually accurate and high-fidelity novel views.

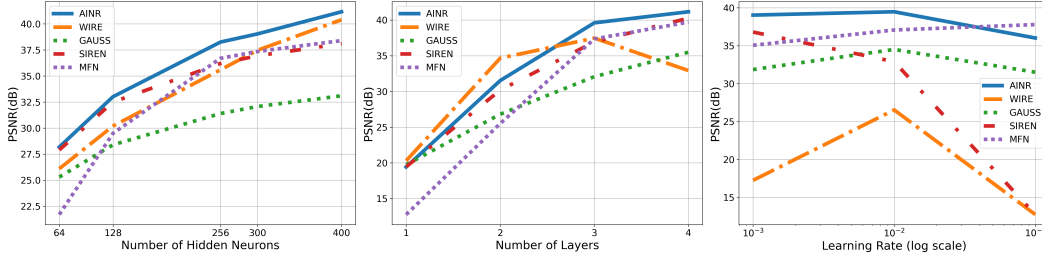


Figure 20: Effect of the MLP architecture’s hyperparameters on AINR’s performance

#### A.6.2 EFFECT OF WEIGHT INITIALIZATION

In most recent INR literature, it has been shown that INRs which are equipped with space-frequency compact activation functions performance do not degrade with the weight initialization mechanism unlike sinusoidal activations (Ramasinghe & Lucey, 2022; Saragadam et al., 2023), . All the presented experimental results in this study is obtained through Pytorch’s default weight initialization scheme. In this weight initialization process, the standard deviation stdv is defined as the reciprocal of the square root of the number of input units  $n_{in}$ . Specifically,  $stdv = \frac{1}{\sqrt{n_{in}}}$ , where  $n_{in}$  represents the number of features or input dimensions in the linear layer. The weights of the layer are then initialized by drawing each element of the weight matrix  $W$  from a uniform distribution in the range  $[-stdv, stdv]$ , i.e.,

$$W_{ij} \sim \mathcal{U}\left(-\frac{1}{\sqrt{n_{in}}}, \frac{1}{\sqrt{n_{in}}}\right)$$

This initialization strategy helps in keeping the weights small enough to ensure stable learning by preventing large gradients during the early stages of training. However, as different weight initializations often lead to different training dynamics of the network, we checked AINR’s performance and the obtained activation sequences for different weight initialization schemes. For this experiment we utilized the Parrot image in the main paper. The obtained results are shown in table 4.

Table 4: Comparison of weight initialization mechanisms

Weight Initialization Mechanism	Sequence	PSNR (dB)
Pytorch Default	PSWF, Gaussian, Gabor	39.04
Xavier Normal	PSWF, Gaussian, Gaussian	37.55
Orthogonal	RRC, Gaussian, Gaussian	37.15
SIREN-like ( $\alpha \neq 30$ )	PSWF, PSWF, Gaussian	36.57
SIREN-like ( $\alpha = 30$ )	PSWF, Sinc, PSWF	36.73

As shown in table 4, the sequence of activations changes when the weight initialization mechanism of the network is altered. This can be attributed to the fact that AINR optimizes the network starting from the given weight initialization scheme, attempting to navigate the loss landscape toward a local minimum. When the distribution of initial weights is significantly changed, it affects the optimization trajectory. The network’s parameters are adjusted based on gradients calculated from this starting point, and the training dynamics follow different paths depending on the initial conditions. Since the loss landscape may contain multiple local minima, the optimization path taken by the network can vary. This explains why different activation function sequences are optimal for different weight initializations—each initialization leads the network to explore a different part of the loss landscape, and the network adapts its activations accordingly to minimize the loss. Therefore, due to the variability in the loss landscape and the dependence on the starting point  $W_0$ , the training

process may converge to different local minima, resulting in variations in the matched activation sequences for each weight initialization scheme. Therefore, it can be concluded that PyTorch's default weight initialization is the most effective for *AINR*.

#### A.7 EFFECT OF POSITIONAL ENCODING

Positional embedding schemes are often utilized in INRs as a method of coordinate transformation. This technique functions similar to a frequency modulation mechanism, enabling the embedding of high frequencies within the signal. In our study, we assessed the performance of *AINR* when incorporating this coordinate transformation.

The left side of figure 21 displays the decoded representations obtained from *AINR* with positional embedding, while the right side shows those from *AINR* without the positional embedding. Given that *AINR* adopts a sequential training approach as outlined in algorithm 1, it allows for the observation of the decoded image at each layer. These images, referred to as 'Layer 1', 'Layer 2', and 'Layer 3' in figure 21, demonstrate the outcomes at each stage.

The comparison reveals that the integration of positional embedding within *AINR* results in improved performance for both the first and second layers, in contrast to the standard *AINR* without any positional embedding. However, upon determining the best activations for the initial two layers, the introduction of the third layer shows that the standard *AINR*, devoid of positional embedding, optimizes the MLP in a manner that it outperforms the *AINR* with positional embedding.

Consequently, these findings suggest that when the optimal sequence of activations is identified for an INR that does not utilize positional embeddings, it can achieve better outcomes than an INR which relies on positional embedding to discover the optimal sequence of activations. This insight showcases the potential for INRs to encode high-frequency components effectively, even in the absence of positional embedding mechanisms, by meticulously selecting the sequence of activations.

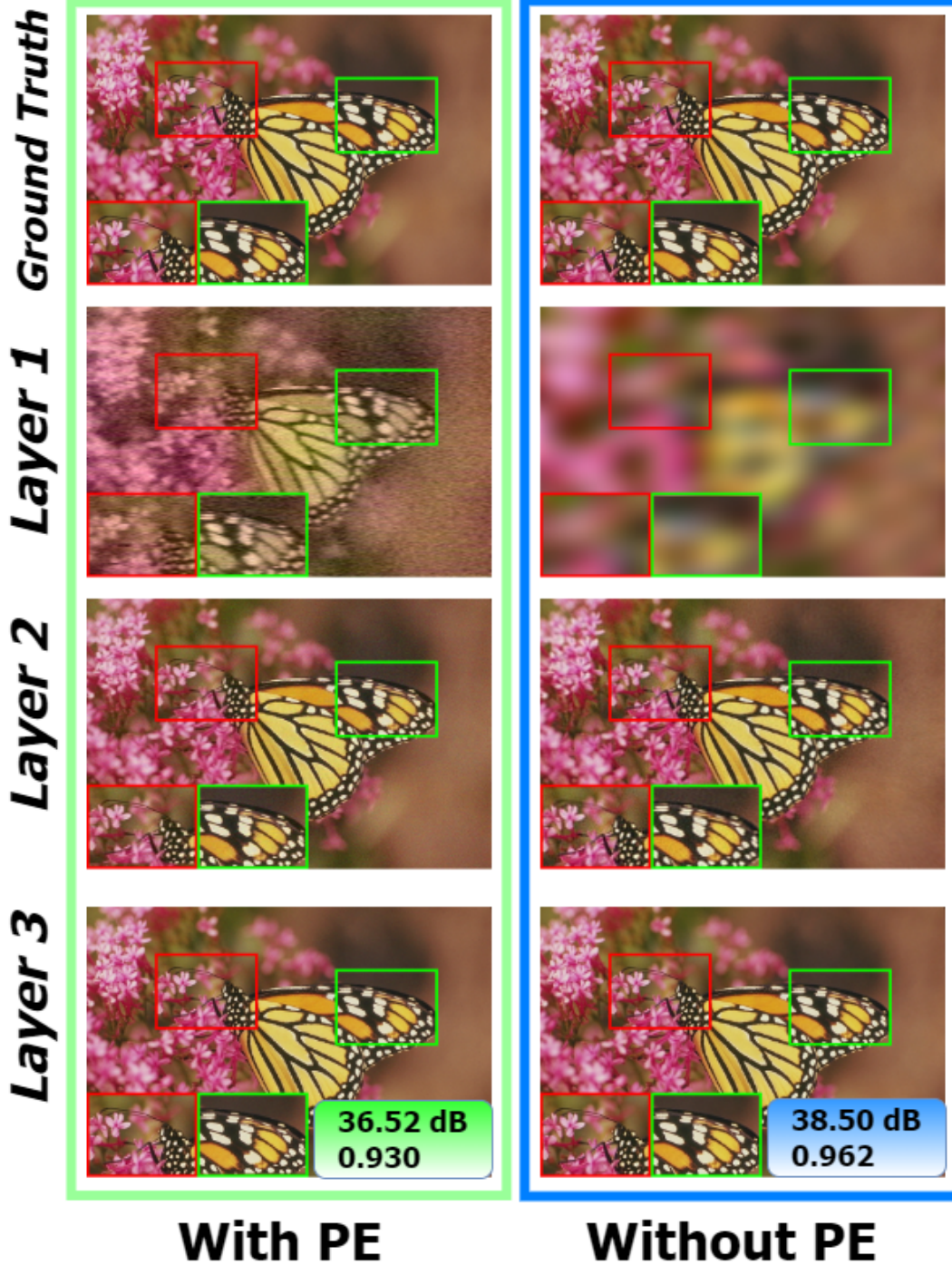


Figure 21: **Effect of positional embedding on AINR**: The first column displays images obtained under AINR’s sequential training with positional embedding, while the second column depicts results without using positional embedding. It is evident that for the first two layers, AINR with positional embedding scheme yields better results compared to the AINR without positional embedding. However, upon introducing the third layer, AINR without positional embedding surpasses the AINR with positional embedding