SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — MATHEMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Mathematics in Data Science

Neural Network Inductive Biases for High-fidelity 3D Reconstruction

Oleg Magnes

SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — MATHEMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Mathematics in Data Science

Neural Network Inductive Biases for High-fidelity 3D Reconstruction

Author:	Oleg Magnes
Examiner:	Prof. Dr. Angela Dai
Supervisor:	Prof. Dr. Angela Dai
Submission Date:	14 July 2025

Acknowledgments

I would like to express my gratitude to Prof. Dr. Angela Dai for the guidance, constructive feedback, and constant support.

Abstract

High-fidelity 3D reconstruction demands representations that capture both coarse structure and subtle geometric details. Implicit Neural Representations can encode 3D geometry at arbitrary resolution, yet their accuracy is limited by the inductive biases of coordinate multilayer perceptrons, which favor low-frequency over high-frequency content. This thesis analyses biases for ReLU-based and sine-based implicit neural representations and proposes learnable sinusoidal activations that extend fixed-frequency SIREN layers. By letting the network tune amplitudes, frequencies, and phase shift of its activation functions depending on the input coordinate, the network can recover finer details and achieve lower reconstruction error relative to SIREN at a comparable parameter count, while also being more robust to the initial hyperparameters.

Contents

1 Introduction

High-fidelity 3D reconstruction is a fundamental goal in computer vision and graphics, involving the creation of detailed models of real-world objects or scenes. Achieving high fidelity requires capturing fine geometric details, which is something conventional representations typically accomplish only at very high resolutions. Conventional discrete representations (such as voxel grids, point clouds, or meshes with fixed resolutions) face a trade-off between detail and memory: to preserve more detail, the representation's size grows significantly. In recent years, Implicit Neural Representations (INRs) have emerged as a powerful alternative for representing 3D objects and other signals in a continuous manner. An INR is typically a neural network that takes a coordinate as input (for example, a spatial (x, y, z) location in a scene) and outputs a signal value at that location (for example, a signed distance or occupancy). Unlike a voxel grid, which is defined only at discrete points, an INR defines the signal for every coordinate in a continuous domain and, therefore, arbitrary resolution, which is restricted only by the capacity of the trained network itself.

Despite these advantages, the performance of INRs and neural networks in general is influenced by their inductive biases. Inductive bias refers to a predisposition of the model to prefer certain kinds of solutions or functions over others [1]. They could arise from the model's architecture, initialization, training procedure, and other design choices. For example, a fully-connected network's depth or activation functions impose inductive biases: some network architectures inherently favor smooth or simple functions, while others can represent more oscillatory or complex patterns.

Inductive biases can help the neural networks to generalize, but they can also limit a model's ability to fit very high-frequency signals. For example, simplicity bias is commonly regarded as key to the success of ReLU-based networks, but at the same time might not be optimal for specific tasks [2] like tabular data or regression tasks. Another related concept is a spectral bias, which is a frequency-domain preference of many neural networks to fit the low-frequency (smooth, slowly varying) components first, and only later the high-frequency components. [3]. Understanding of how inductive biases affect the performance of INRs can help to create more powerful networks that can capture both high-frequency and low-frequency signals simultaneously. The papers like SIREN [4] or FINER [5] partially weaken the low-frequency spectral bias of coordinate multilayer perceptrons (MLPs) and allow for better capture of high-frequency details.

This work [1] continues the research of FINER and STAF [6] that allows for more flexible activation functions, which can ease the spectral bias even further. We show that allowing the network to learn amplitude, frequency, and phase shift coefficients for its activation functions jointly with the task enables our model to outperform SIREN. Coefficients are predicted based on the input coordinate, allowing the model to use different activation functions on distinct scene regions. Finally, we demonstrated that our solution is more robust to initial hyperparameters than SIREN.

[1]Code is available on GitHub: `https://github.com/ttaggg/inductive_biases`

2 Related Work

2.1 Overview of the section

This section starts with the paper Neural Redshift [1] that describes inductive biases of the neural networks and that provides ideas that might be useful for training implicit neural representations. There are several important papers discussing various implicit neural representations architectures and how to improve them. The groundbreaking SIREN [4] paper was the first to suggest periodic functions as an activation function for multilayer perceptrons (MLP) used to encode various signals. The papers FINER [5], STAF [6], and CAM [7] describe methods to improve the performance of SIREN by adjusting parameters of the activation function. Finally, there are papers that describe different views on approaches to fit more complex signals: by using a hybrid implicit-explicit architecture (ACORN [8]), or by representing signals on multiple scales (BACON [9]).

2.2 Neural Redshift: Random Networks are not Random Functions

This work [1] describes inductive biases of neural networks. Inductive biases are assumptions about the target function encoded in the learning algorithm as the hypothesis class (e.g., architecture), objective (e.g., loss function), and optimization method (e.g., SGD). Traditionally, the generalization ability of neural networks has been mainly attributed to implicit biases introduced by optimization algorithms, however, this work concentrates on inductive biases that arise from the architecture itself, independent of any training. Authors show that even randomly initialized networks exhibit non-random biases that occur when changing the depth or width of the network, using different activations, or using skip-connections.

The authors perform experiments involving randomly initialized (i.e., untrained) neural networks. These networks are tested by evaluating their behavior over a grid of input points, producing an output signal that is then analyzed for complexity. The methodology overview is shown in Figure 2.1.

The main findings were that neural networks are biased in implementing functions of

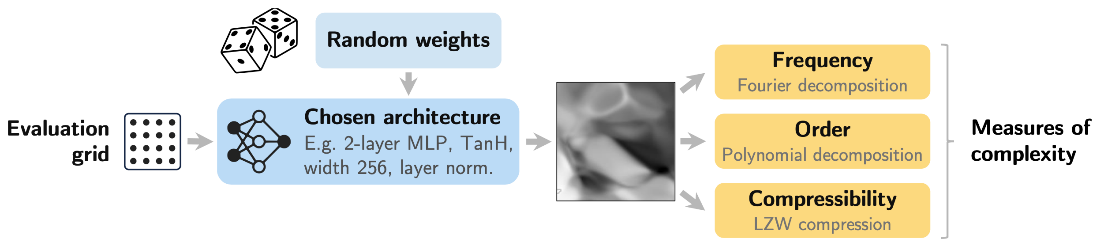


Figure 2.1: Methodology overview from the Neural Redshift paper showing how complexity metrics are applied to analyze network behavior, from [1].

a particular level of complexity (not necessarily low), as determined by the architecture. Simplicity bias is not universal but depends on standard components (ReLUs, residual connections, layer normalizations). ReLU networks also have the unique property of maintaining their simplicity bias for any depth and weight magnitudes. The table 2.1 contains the overview of the effect that different architectural decisions have on the complexity of the output.

Table 2.1: Components that bias NNs towards low/high complexity.

Lower complexity	**No impact**	**Higher complexity**
ReLU-like activations	Width	Other activations
Small weights / activations	Bias magnitudes	Large weights / activations
Layer normalization		Depth
Residual connections		Multiplicative interactions

Additionally authors showed the practical impacts of inductive biases on two tasks.

- In a binary classification task involving modulo addition networks with ReLU activations failed to generalize, despite fitting the training data perfectly. In contrast, architectures with tunable complexity (using Gaussian or sine activations) were able to generalize successfully, but only when their complexity matched that of the target function.
- Shortcut learning: in the regression tasks similar to Colored-MNIST, models need to predict a score in [0, 1] proportional to the digit value as well as to the color intensity. Authors observed conditions when models were biased toward simplicity and preferred to rely on the color channel (a simpler feature) rather than the digit (a more complex feature). The authors showed that the best results were achieved by the models having intermediate complexity on the initialization

and claimed that it was due to the match between the complexity of the model and the complexity of the target function.

These results suggest that there is a strong correlation between the complexity at initialization and in the trained model. Therefore, inductive bias at initialization can either help or harm generalization, depending on how well the model's complexity matches the complexity of the modeled data.

2.3 Implicit Neural Representations with Periodic Activation Functions

A significant advancement in the field of implicit neural representations is the introduction of Sinusoidal Representation Networks (SIRENs) [4]. This work proposes a novel architectural paradigm for representing continuous signals using neural networks that employ periodic activation functions, particularly the sine function. Traditional implicit neural representations use multilayer perceptrons (MLPs) with ReLU or similar activations. However, these networks struggle with encoding high-frequency signals. In this paper, the authors investigated the performance of implicit neural representations with the sine activation function and also suggested a weight initialization scheme that improves the convergence.

In this formulation the layer of SIREN is a dense layer with a sine nonlinearity applied element-wise:

$$\mathbf{x}_i \mapsto \phi_i(\mathbf{x}_i) = \sin\left(\omega_i \cdot (\mathbf{W}_i \mathbf{x}_i + \mathbf{b}_i)\right)$$

where $\mathbf{W}_i \in \mathbb{R}^{N_i \times M_i}$, $\mathbf{b}_i \in \mathbb{R}^{N_i} \in$ are weight matrix and bias of the i-th layer and $\omega_i \in \mathbb{R}$ is a hyperparameter frequency scaling factor.

Authors suggest the following initialization scheme: wights are drawn from the $w_i \sim \mathcal{U}\left(-\sqrt{c/n}, \sqrt{c/n}\right)$, where c is a constant and n is the number of inputs for each neuron. The value $c = 6$ ensures that the input to each sine activation is normally distributed with a standard deviation of 1. Since only a few weights have a magnitude larger than π, the frequency throughout the sine network grows slowly. In practice $w_i \sim \mathcal{U}\left(-\sqrt{1/n}, \sqrt{1/n}\right)$ for the first layer and $w_i \sim \mathcal{U}\left(-\frac{\sqrt{c/n}}{\omega_i}, \frac{\sqrt{c/n}}{\omega_i}\right)$ for all the others, where ω_i in the denominator is a frequency scaling factor in the corresponding layer.

The authors demonstrated superior results on image fitting, 3D shape fitting, audio fitting, and solving PDEs compared to baselines that use TanH and ReLU as activation functions. Overall, by demonstrating that sine-based networks are not only expressive but also trainable with principled initialization, the authors provide a general-purpose tool for a variety of scientific and machine learning problems. Main limitation of

SIRENs, however, is that it requires a very careful initialization to achieve optimal performance. Other approaches, such as FINER [5] and BACON [9], are trying to solve this problem.

2.4 FINER: Flexible spectral-bias tuning in Implicit Neural Representation by Variable-periodic Activation Functions

The FINER paper [5] is based on two ideas:

- Conventional coordinate-MLPs are prone to a spectral bias, they tend to learn low-frequency components first and to under-represent high-frequency detail. Fourier Features [10] and SIREN [4] can alleviate this issue, however, the former must have a predefined basis' frequency set carefully chosen to match the complexity of the data, and the latter requires a careful initialization.
- Capacity-convergence gap of SIREN, meaning that the capacity of SIREN is limited by the choice of ω_0.

Variable-periodic activation functions can properly address these problems. The suggested activation function is

$$\sigma(\mathbf{x}) = \sin((|\mathbf{x}| + 1) \cdot \mathbf{x})$$

so that the local frequency increases with the magnitude of the input.

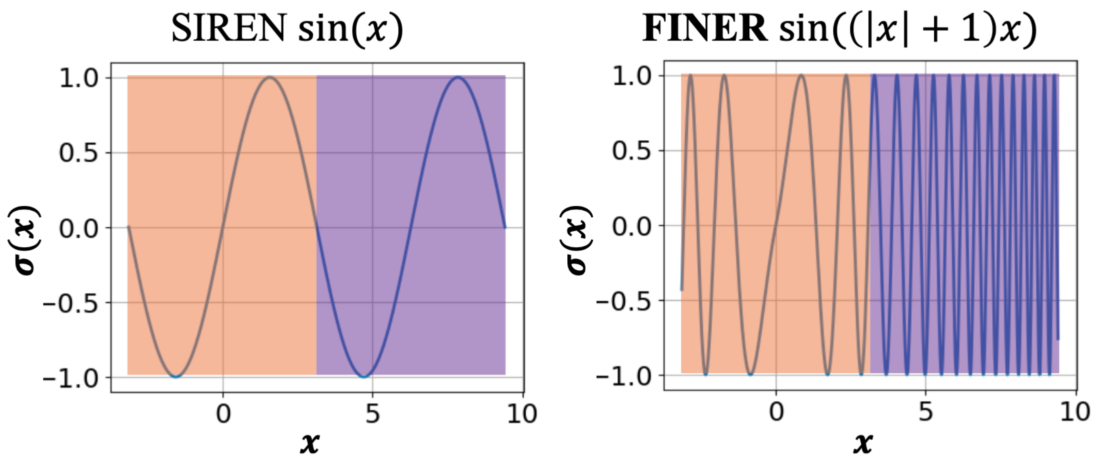


Figure 2.2: Comparison between SIREN and FINER activations, from [5].

The authors claim that by initializing the bias of the neural network within different ranges, sub-functions with various frequencies in the variable-periodic function are selected for activation. However, in practice, weight initialization is often the same as SIREN's, and the full activation function contains ω_i as well (default value is 30, same as SIREN). In this setting, the activation function still allows to dynamically alter the local frequency.

$$\mathbf{y}_i := \mathbf{W}_i\mathbf{x}_i + \mathbf{b}_i$$
$$\mathbf{x}_i \mapsto \sigma(\mathbf{x}_i) = \sin\left(\omega_i \cdot ||\mathbf{y}_i + 1|| \cdot \mathbf{y}_i\right)$$

The authors tested FINER on various tasks, such as image fitting, 3D shapes fitting, and NeRF optimization. FINER, on average, outperforms other architectures like SIREN, Wire, ReLU-PE, and Gaussian activation functions. A follow-up work, FINER++ [11], shows that the variable-periodic trick generalises beyond sine to Gaussian and wavelet bases.

2.5 Coordinate-aware modulation for Neural Fields

The authors of CAM [7] suggest a method that injects spectral bias-free grid representations into the intermediate features in the MLP. CAM modulates the intermediate features of the neural networks based on input coordinates, where the scalar factors for scale and shift modulation are parameterized by grids:

$$\widetilde{F}_{n,c} = \gamma_n(X;\Gamma)F_{n,c} + \beta_n(X;B), \tag{2.1}$$

where $F, \widetilde{F} \in \mathbb{R}^{N\times C}$ are an intermediate feature tensor and the modulated output feature, N is a batch size, C is a channel size, and n, c denote the batch and channel index of the feature, respectively. $\gamma(\cdot;\Gamma), \beta(\cdot;B) : \mathbb{R}^{N\times D} \to \mathbb{R}^{N}$ are the scale and shift function of input coordinates $X \in \mathbb{R}^{N\times D}$. $\gamma_n(\cdot;\Gamma), \beta_n(\cdot;B)$ denote scale and shift factor for batch index n.

Importantly, CAM incorporates feature normalization prior to modulation, which was previously ineffective in neural fields but is shown here to improve convergence stability and representation performance significantly.

The authors showed that coordinate-aware modulation consistently improves the performance of FFN [10] on the image fitting task, although there is no direct comparison with SIREN or SIREN using coordinate-aware modulation. Also, there are no results for 3D shape fitting that would be important in the context of this work.

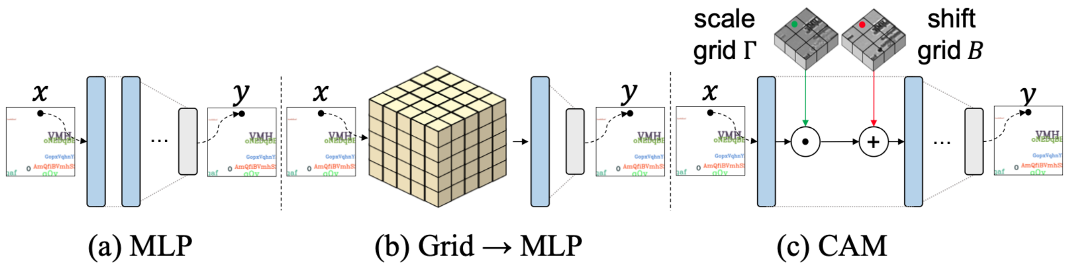


Figure 2.3: Comparison between methods (a) MLP, (b) Grid -> MLP, (c) CAM, from [7].

2.6 STAF: Sinusoidal Trainable Activation Functions for Implicit Neural Representation

In this paper [6], the authors suggest a method that generalizes periodic activation functions like SIREN, and at the same time propose a novel initialization scheme.

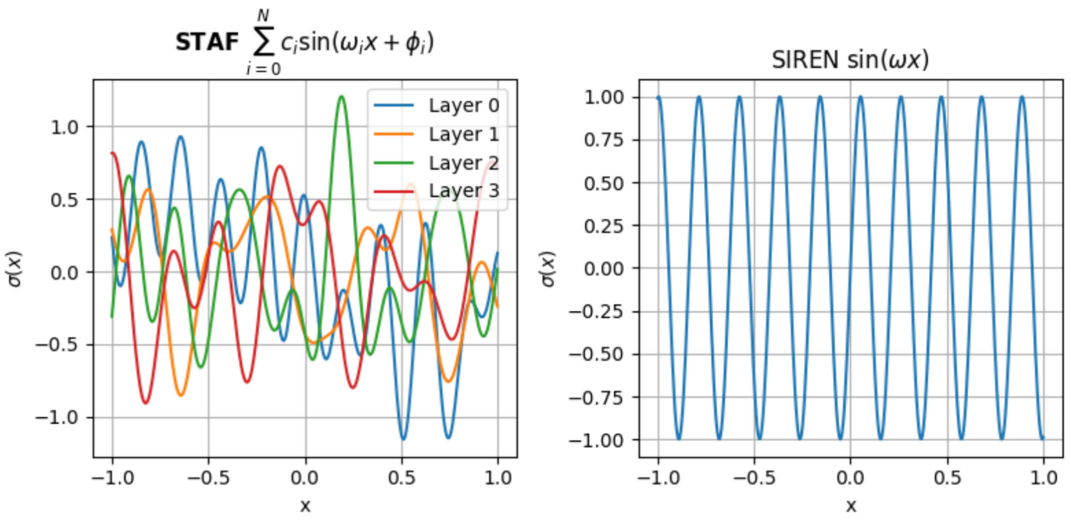


Figure 2.4: Comparison between STAF and SIREN activation function, from [6].

STAF activation function is:

$$\sigma(x) = \sum_{i=1}^{\tau} C_i \sin(\Omega_i x + \Phi_i),$$

where C_i is the amplitude, Ω_i is a frequency, and Φ_i is a phase parameters. These parameters are learned during the training together with the main objective.

The authors describe three possible implementation strategies:

- Individual Neuron Activation: assign a unique activation function to each neuron.
- Uniform Network-wide Activation: a single shared activation function is used across the entire network.
- Layer-wise Shared Activation: use a distinct shared activation function for each layer. This method is used for all experiments in the paper.

According to the provided results, STAF convincingly outperforms all compared models on various tasks, including image and 3D shape fitting. Not only does STAF outperform methods like SIREN and FFN, but also more recent ideas like FINER and INCODE [12].

2.7 Modulated Periodic Activations for Generalizable Local Functional Representations

This paper presents [13] another approach to modulate amplitude, frequency, and phase shift using a dual MLP setup. Prior works typically fit a separate Multi-Layer Perceptron (MLP) to each signal, often using ReLU activations, yielding good accuracy but no generalization. On the other hand, SIREN can represent high-frequency details, but it is also usually trained per sample. The authors of [13] propose a dual-MLP architecture (a synthesis network and a modulator network) called ModSiren.

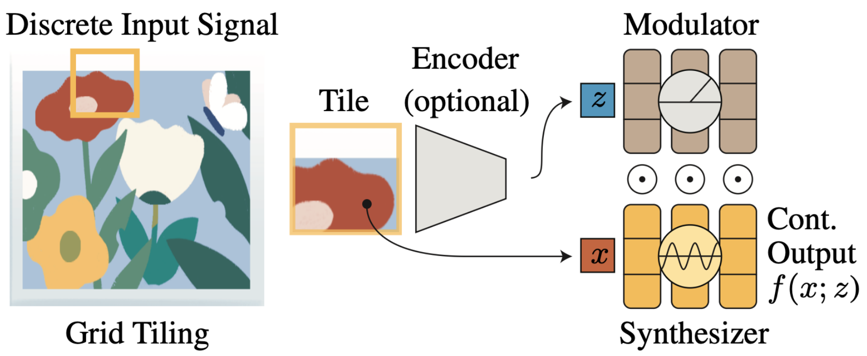


Figure 2.5: Encoding discrete signals as neural-functional representations, from [13].

The synthesis network is MLP with sine activations that maps coordinate to the RGB-color or SDF scalar

$$h_i = \alpha_i \odot \sin(w_i h_{i-1} + b_i)$$

where $w_i \in \mathbb{R}^{d_i \times d_{i-1}}$ and $b_i \in \mathbb{R}^{d_i}$ are trainable weights and biases for a corresponding layer, and $\alpha_i \in \mathbb{R}^{d_i}$ is a modulation variable returned by the modulator and $\odot$ is element-wise multiplication.

The modulator network acts on the latent code z that corresponds to a target signal.

$$h'_0 = \mathrm{ReLU}(w'_0 z + b'_0)$$
$$\alpha_{i+1} = h'_{i+1} = \mathrm{ReLU}(w'_{i+1}[h'_i \quad z]^T + b'_{i+1})$$

For each layer α_i modulates the activation functions of the synthesis network:

$$h_i = \alpha_i \odot \sin(w_i \cdot (\alpha_{i-1} \odot \sin(\cdots)) + b_i)$$

$alpha_i$ directly affects amplitudes, and also indirectly control the frequency and phase shift of the sinusoids in subsequent layers. This allows fitting the whole datasets into one synthesis network modulated by latent codes of samples (learned jointly or obtained with an encoder) or to fit large images by splitting them into tiles, each having a latent code.

2.8 ACORN: Adaptive Coordinate Networks for Neural Scene Representation

The task of modeling large-scale and high-resolution objects, such as very large images (gigapixel size) or 3D scenes (> 300k polygons), could be challenging. Authors of the ACORN paper [8] model the problem with an implicit-explicit network architecture that adaptively allocates resources during training and inference based on the local complexity of the signal.

Overall signal is modeled by a coordinate network that takes a multiscale block coordinate as an input and returns a quantity of interest continuously within a block. The architecture consists of:

- Coordinate encoder: input coordinate is not mapped directly to the output, but to a low-resolution grid of local features.

$$\Phi : \mathbb{R}^{d_{\text{in}}+1} \to \mathbb{R}^{C \times N_1 \times \cdots \times N_{d_{\text{in}}}}, \quad \mathbf{x}_g^i \mapsto \Phi(\mathbf{x}_g^i) = \Gamma^i$$

- Feature decoder: interpolate features and decode them to evaluate the representation at any continuous coordinate within the block.

$$\Psi : \mathbb{R}^C \to \mathbb{R}, \quad \gamma^i_{\mathbf{x}_1} \mapsto \Psi(\gamma^i_{\mathbf{x}_1}) = y^i \approx y^i_{\text{GT}}$$

The feature grid generation is done only once per block by the coordinate encoder, which is fast and memory efficient. Additionally, authors suggested a training strategy that adaptively refines the scale and location of block coordinates during training, similar to octree / quadtree. Optimal block decomposition is an integer linear program optimization.

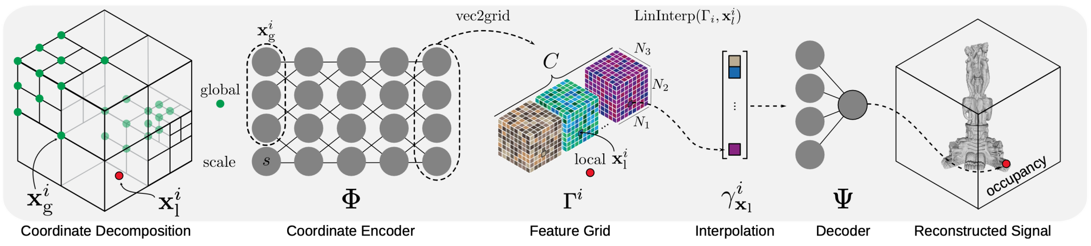


Figure 2.6: Illustration of multiscale network architecture for 3D shape representation, from [8].

The authors showed that their suggested architecture allows a significant improvement in scale: training time required for a model and required GPU memory are smaller compared to SIREN or ReLU-PE, while outperforming these models on the image and 3D reconstruction tasks. The paper has its limitations, for example, ACORN does not explicitly enforce continuity of the signal across blocks, and therefore, artefacts near the boundaries could happen. Updating the partitioning requires retraining on new blocks, so periodic spikes in loss are expected. Also, the optimization of block decompositions relies on an Integer Linear Program (ILP), which is non-differentiable and therefore makes convergence slower.

2.9 BACON: Band-limited Coordinate Networks for Multiscale Scene Representation

BACON (Band-limited Coordinate Networks) [9] introduces a new coordinate-based neural architecture for signal representation that is analytically band-limited. It builds upon Multiplicative Filter Networks (MFNs) [14], which differ from conventional MLPs in that they employ a Hadamard product between linear layers and sine activation functions. The authors extend this architecture to control the frequency content analytically through fixed-bandwidth sine layers to achieve multiscale, band-limited outputs.

The architeture is shown on the figure 2.7. Input coordinate $\mathbf{x} \in \mathbb{R}^{d_{in}}$ is passed through several layers of the form $g_i : \mathbb{R}^{d_{\text{in}}} \mapsto \mathbb{R}^{d_h}$ with $g_i(\mathbf{x}) = \sin(\boldsymbol{\omega}_i \mathbf{x} + \boldsymbol{\phi}_i)$, for $i = 0, ..., N_{L-1}$, and N_L is the number of layers in the network. Intermediate output of

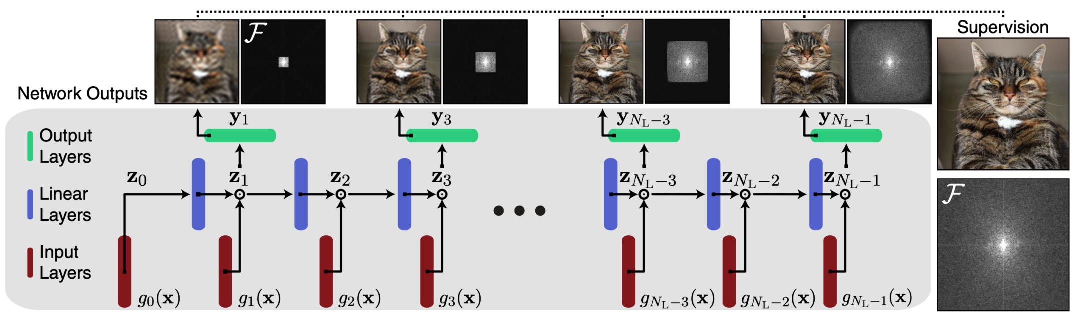


Figure 2.7: Overview of BACON architecture, from [9].

the network $\mathbf{y}_i \in \mathbb{R}^{d_{out}}$ at the i-th layer is defined the following way:

$$\begin{aligned}
\mathbf{z}_0 &= g_0(\mathbf{x}) \\
\mathbf{z}_i &= g_i(\mathbf{x}) \circ (\mathbf{W}_i \mathbf{z}_{i-1} + \mathbf{b}_i), \quad 0 \le i < N_L \\
\mathbf{y}_i &= \mathbf{W}_i^{\text{out}} \mathbf{z}_i + \mathbf{b}_i^{\text{out}}
\end{aligned}$$

where $\circ$ is a Hadamard product.

The output can be expressed as a sum of sines with varying amplitude, frequency and phase shift.

$$\mathbf{y}_i = \sum_{j=0}^{N_{\text{sine}}^{(i)}-1} \bar{\alpha}_j \sin(\bar{\omega}_j \mathbf{x} + \bar{\phi}_j)$$

The frequencies ω_i are frozen and not optimized. The authors set them to a bandwidth in $[-B_i, B_i]$ using random uniform initialization. Then the total bandwidth of an output at layer i of the network is less than or equal to $\sum_{j=0}^{i} B_j$ and the maximal bandwidth is $B = \sum_{i=0}^{N_L-1} B_i$. This way, unlike MLP-based architectures whose frequency content is difficult to predict or constrain, BACON allows each layer's maximum frequency to be set at initialization.

BACON shows good results on the image fitting, neural rendering, and 3D shape fitting tasks. For 3D shape fitting, the authors fit the SDF volume and compare it with various architectures. BACON falls second after the Fourier Features [10] but outperforms SIREN.

Also, the authors suggest a way to accelerate the performance of the network when evaluating SDFs on a dense grid for mesh extraction by stopping the forward pass early if the current input coordinate is far from the surface according to the intermediate layers (i.e., no need to refine further).

3 Methods

3.1 Dataset

The dataset used in this work is the Scannet++ Dataset [15], which is a large-scale dataset that contains high-detail 3D indoor scenes. For this work, parts of the scene that contain both high-frequency and low-frequency details were extracted.

- Points with normals were used as the input data.
- Meshes from the datasets were used for LPIPS evaluation.
- Labels for mesh vertices were extended to the whole point cloud using the label of the nearest neighbor and were also used to evaluate high-frequency and low-frequency regions separately.

Two scenes from Scannet++ are used for experiments: id 8b5caf3398, called *office room* in this thesis (∼4.5 million points), and id 5fb5d2dbf2, called *machinery room* (∼9.1 million points). For additional tests and sanity checks, the Thai Statue from the Stanford 3D Scanning Repository was used.

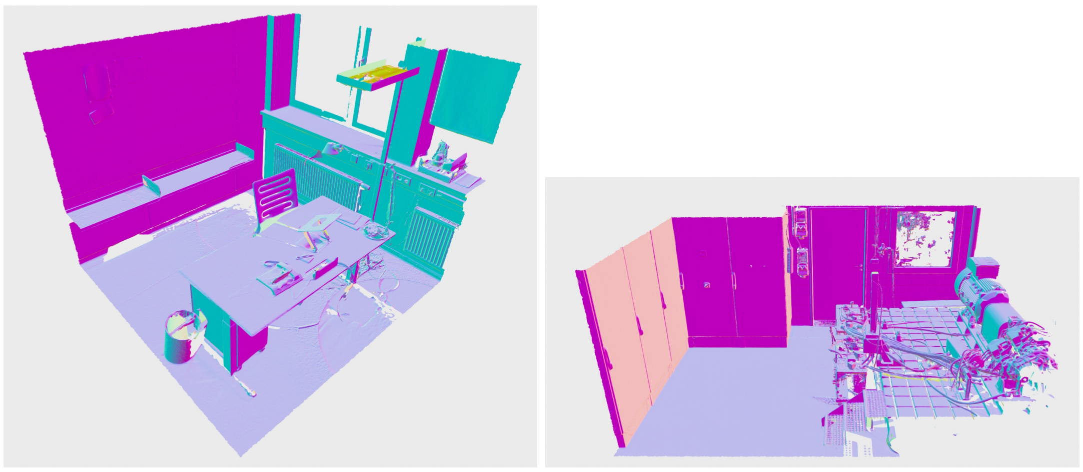

(a) 8b5caf3398 (b) 5fb5d2dbf2

Figure 3.1: Scenes from Scannet++.

3.2 Training procedure

3.2.1 Input data

We follow the training procedure for SIREN [4]. The scene is normalized to be in the range of $[-1, 1]^3$. For each batch, half of the points come from the point cloud (and are denoted as on-surface points), and the remaining half is uniformly sampled from $[-1, 1]^3$ range and denoted as off-surface points.

3.2.2 Loss function

For most of the experiments, the loss function follows the loss function from the SIREN. Values for λ_i were modified to get the best result.

$$\mathcal{L} = \lambda_1 \mathcal{L}_{Eikonal} + \lambda_2 \mathcal{L}_{SDF} + \lambda_3 \mathcal{L}_{normal} + \lambda_4 \mathcal{L}_{off-surface}$$

$$\begin{aligned}
\mathcal{L}_{Eikonal} &= \int_{\Omega} \| \|\nabla_{\mathbf{x}} \Phi(\mathbf{x})\| - 1 \| \, d\mathbf{x} \\
\mathcal{L}_{SDF} &= \int_{\Omega_0} \|\Phi(\mathbf{x})\| \, d\mathbf{x} \\
\mathcal{L}_{normal} &= \int_{\Omega_0} (1 - \langle \nabla_{\mathbf{x}} \Phi(\mathbf{x}), \mathbf{n}(\mathbf{x}) \rangle) \, d\mathbf{x} \\
\mathcal{L}_{off-surface} &= \int_{\Omega \setminus \Omega_0} \psi(\Phi(\mathbf{x})) \, d\mathbf{x}
\end{aligned}$$

Where $\Phi(\mathbf{x})$ is the model, $\psi(\mathbf{x}) = \exp(-\alpha \cdot |\Phi(\mathbf{x})|), \quad \alpha \gg 1$, Ω is the whole domain and Ω_0 is the zero set.

Eikonal constraint $\mathcal{L}_{Eikonal}$ pushes the gradient norm to one, SDF constraint $\mathcal{L}_{SDF}$ penalizes on-surface points that have nonzero SDF values, normal constraint $\mathcal{L}_{normal}$ makes sure the gradient has the same direction as normals, and off-surface constraint $\mathcal{L}_{off-surface}$ penalizes small predicted SDF values in off-surface points.

Coefficients for loss terms are: $\lambda_1 = 5 \cdot 10^1$, $\lambda_2 = 3 \cdot 10^3$, $\lambda_3 = 1 \cdot 10^2$, $\lambda_4 \in [100, 500]$ depending on the model's performance and $\alpha = 1 \cdot 10^2$.

3.2.3 Optimization

All models were trained using the Adam optimizer. Learning rate followed cosine annealing scheduling with initial learning rate being $1 \cdot 10^{-4}$. The models were trained for 3000 epochs. The batch size for every model was 100,000.

For the training involving the modulator (described in the section 3.4), the modulator was overfitted to predict $\alpha_i = 1.0$, $\beta_i = 1.0$, and $\gamma_i = 0.0$ prior to the training, and this overfitted version was used for initialization. This approach showed better stability in the early stages of the training without harm to accuracy.

3.2.4 Outputs

All models are functions of the form $\Phi(\mathbf{x}) : \mathbb{R}^3 \mapsto \mathbb{R}$, and predict one scalar value, SDF (signed distance function / field). SDFs can be used to describe geometrical objects. The absolute value this function takes on some input coordinate is the distance to the closest surface, and the sign of the value denotes whether the coordinate is inside or outside of the object. For SDF $|\nabla f| = 1$, the gradient satisfies the eikonal equation and on the boundary $\nabla f(\mathbf{x}) = N(\mathbf{x})$, where $N(\mathbf{x})$ is a normal vector field.

3.3 Evaluation metrics

3.3.1 Preprocessing

- The models are evaluated on the dense grid of the size $1536 \times 1536 \times 1536$.
- The marching cubes algorithm is applied to the resulting 3D SDF volume to obtain the mesh.
- N = |GT point cloud| points are sampled from the faces of the mesh with the probability proportional to the face's area.

3.3.2 Chamfer distance

Given two point clouds P and Q, the Chamfer distance is defined as the sum of two directional distances:

$$\mathrm{CD}(P,Q) = \frac{1}{|P|} \sum_{p \in P} \min_{q \in Q} \|p - q\|_2^2 + \frac{1}{|Q|} \sum_{q \in Q} \min_{p \in P} \|q - p\|_2^2$$

Due to the nature of how SDF works for non-watertight meshes, it was common to get artifacts like additional surfaces towards the boundary of the scene after Marching Cubes was applied to the SDF volume. These artifacts on the predicted point cloud can skew the evaluation metric significantly. To get a better understanding of how well the target scene was reconstructed, only the distances from the target to the predicted point cloud are given:

$$\overrightarrow{CD}(T,P) = \frac{1}{|T|} \sum_{t \in T} \min_{p \in P} \|t - p\|_2^2$$

where T, t refers to the target point cloud (ground truth) and P, p to predicted point cloud.

3.3.3 Completeness

The completeness metric measures the ratio of target point cloud points that have a predicted point cloud point in proximity within a given radius r. This metric has benefits over Chamfer distance due to its robustness to outliers. It is defined as:

$$\text{Completeness}(P_{GT}, P_{pred}, r) = \frac{1}{|P_{GT}|} \sum_{p \in P_{GT}} \mathbb{I}\left[\min_{q \in P_{pred}} \|p - q\|_2 \le r\right]$$

Radii $r = 0.002$ (0.2% of a scene) and $r = 0.003$ (0.3% of a scene) were used for evaluation.

3.3.4 Learned Perceptual Image Patch Similarity

Learned Perceptual Image Patch Similarity (LPIPS) is a perceptual similarity metric that measures the perceptual difference between two images using deep features from pre-trained convolutional neural networks. Unlike traditional pixel-wise metrics such as PSNR or SSIM, LPIPS leverages learned representations to better align with human perceptual judgments.

For a pair of image patches x and x_0, LPIPS is computed as:

$$\text{LPIPS}(x, x_0) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} ||w_l \odot (y^l_{hw} - y^l_{0hw})||_2^2$$

where y^l and y^l_0 are the normalized activations from layer l of a pre-trained network (typically VGG or AlexNet), w_l are learned linear weights, and H_l, W_l are the spatial dimensions of layer l.

Nine camera views were chosen for each sample to cover most of the scene. Images from the target and predicted meshes are then rendered and used to calculate the metric. The mean of all LPIPS scores is reported.

3.3.5 Fourier frequency based complexity metric

The key idea of this metric is to measure the complexity of the neural network outputs. We evaluate the neural network $f_\theta(\cdot)$ on the evenly-spaced grid of inputs

$\mathbf{X}_{grid} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ restricted to $[-1, +1]^3$ to get an output volume that we later analyze.

First, compute a discrete Fourier transform that approximates f with a weighted sum of sine functions of various frequencies.

$$f(x) := (2\pi)^{d/2} \int \tilde{f}(\mathbf{k}) e^{ik\cdot\mathbf{x}} d\mathbf{k}$$

where $\tilde{f}(\mathbf{k}) := \int f(\mathbf{x}) e^{-i\mathbf{k}\cdot\mathbf{x}} d\mathbf{x}$ is a Fourier transform. The frequency numbers k are regularly spaced, $\{0, 1, 2, ..., K\}$ with the maximum K set according to the Nyquist–Shannon limit of $\mathbf{X}_{grid}$. Then the metric is defined as follows:

$$C_{\text{Fourier}}(f) = \frac{\sum_{k=1}^{K} \tilde{f}(\mathbf{k}) \cdot k}{\sum_{k=1}^{K} \tilde{f}(\mathbf{k})}$$

Smoothly varying functions are likely to mostly involve low-frequency components, and therefore give output a low value to $C_{Fourier}$, while complex functions would have large high-frequency coefficients and a larger CFourier value.

3.4 SIREN-FM (Frequency Modulated SIREN)

3.4.1 Network's architecture

The main idea of the suggested architecture is that high-frequency activation functions are required to learn highly detailed parts of the scene, and lower frequencies can be used for areas without fine details. This could be done by allowing the model to learn the coefficients for the sine activation function, i.e., amplitude, frequency, and phase shift, jointly with the main objective.

The network has two components: (1) the MLP that takes an input coordinate and returns the scalar, its layers have the activation function: $\sigma(\cdot) = \alpha_i \cdot \sin(\beta_i \omega x + \gamma_i)$ and (2) the modulator, a smaller MLP with positional encoding that takes the input coordinate and returns coefficients α_i, β_i, and γ_i for each layer of SIREN that uses non-linearity. ω here is a non-trainable hyperparameter that corresponds to ω in SIREN's activations. The architecture is shown in Figure 3.2.

3.4.2 Key differences from prior work

The main difference from the FINER [5] is that FINER only learns one coefficient, β_i in our notation, and this coefficient is trained as a function of the corresponding linear layer's weight and bias. The activation in FINER is $\sigma(\mathbf{x}) = \sin((|\mathbf{x}| + 1) \cdot \mathbf{x})$, i.e. the frequency multiplier is $\beta_i = \mathbf{x} = |\mathbf{W}_i \cdot \mathbf{X} + \mathbf{b}_i| + 1$.

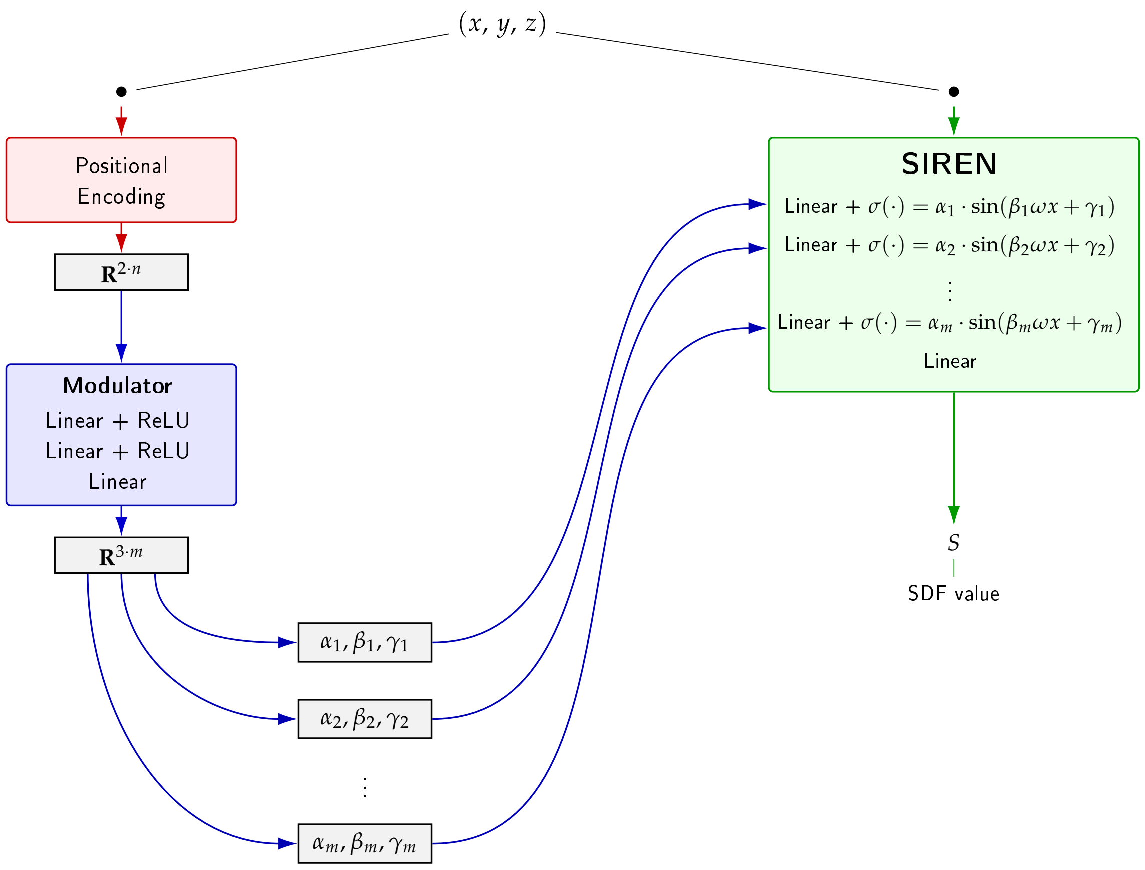

(x, y, z)
Positional Encoding
$\mathbf{R}^{2 \cdot n}$
Modulator
Linear + ReLU
Linear + ReLU
Linear
$\mathbf{R}^{3 \cdot m}$
$\alpha_1, \beta_1, \gamma_1$
$\alpha_2, \beta_2, \gamma_2$
$\alpha_m, \beta_m, \gamma_m$
SIREN
Linear + $\sigma(\cdot) = \alpha_1 \cdot \sin(\beta_1 \omega x + \gamma_1)$
Linear + $\sigma(\cdot) = \alpha_2 \cdot \sin(\beta_2 \omega x + \gamma_2)$
Linear + $\sigma(\cdot) = \alpha_m \cdot \sin(\beta_m \omega x + \gamma_m)$
Linear
S
SDF value

Figure 3.2: The architecture of the network, n is number of frequencies used for positional encoding, m is the number of hidden layers in the SIREN network + 1 (the first layer and all hidden layers are modulated).

The main difference from the STAF [6] approach is that in our case, coefficients are changed dynamically depending on the input, whereas in STAF α_i, β_i and γ_i while being learned parameters, do not change depending on the coordinate. Also, the final activation of each layer in STAF is the sum of sine functions with different coefficients, which should theoretically be more expressive than our approach.

The main difference from the Modulated Periodic Activations for Generalizable Local Functional [13] is that we do not use or learn any latent codes for the respective regions of the 3D scenes. Another difference is that in our approach, we have a single α, β, γ coefficient for each layer's activation instead of a vector of the length $|\#\text{ of neurons in the layer}|$ that only corresponds to a frequency scaling. Also, coefficients are predicted by the network together as an output of the last layer of the modulator. Finally, in the Modulated Periodic Activations, each vector of frequency scaling coefficients is the output of the new dense layer applied to the concatenation of the latent code and frequency scaling vector from the previous layer.

4 Results

4.1 Inductive biases of various architectures

Following the experiments from Neural Redshift [1], we evaluated the complexity of the signal produced by various architectures. Whereas the original study focused on 2D outputs of randomly initialised networks (see 3.3.5), we evaluated $C_{\text{Fourier}}(f)$ on a 3D volume created by sampling the networks over a uniform grid in $[-1, 1]^3$. We sampled the networks' biases from $\mathcal{U}(-1, 1)$ and weights from $\mathcal{U}(-s, s)$, where $s = \sqrt{\alpha/(fan_{in} + fan_{out})}$ and α is a coefficient used to regulate the weights magnitude. $C_{\text{Fourier}}(f)$ was calculated for the 3D volume obtained on the evenly-spaced grid of inputs $[-1, 1]^3$.

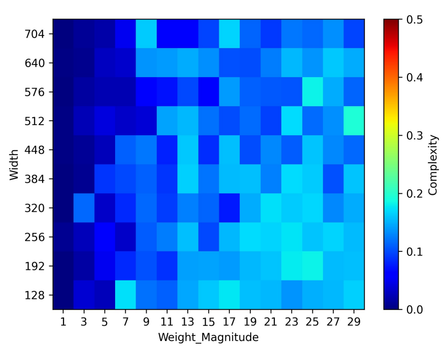
The default values for hyperparameters were: depth = 3 (3 hidden layers), width = 256, weight magnitude (α) = 6.

4.1.1 ReLU and ReLU + PE

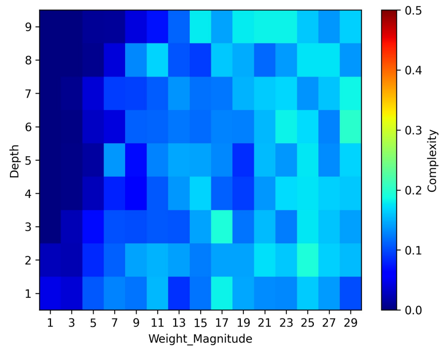
Across all tested depths and widths, ReLU-based networks exhibit low Fourier complexity, although there is an increase in complexity with the increased weight magnitude, which was not highlighted in the original paper. Also, Neural Redshift did not include the positional encoding depicted in Figure 4.2. Here we use a (NeRF-like) positional encoding:

$$\gamma(x) = [sin(\pi x), cos(\pi x), sin(2\pi x), cos(2\pi x), \cdots, sin(2^{L-1}\pi x), cos(2^{L-1}\pi x)]$$

where L is the degree of the positional encoding (default = 10). Note that is our case $\gamma(\cdot)$ is applied to all three axes. Overall, positional encoding seems to compensate for the intrinsic simplicity of ReLU-based INRs and produces a high complexity outputs. As seen on the Figure 4.3 signal gradually becomes more complex with the length of the positional encoding vector.

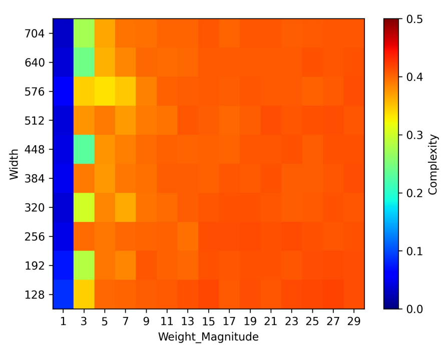


(a) Width and weights' magnitudes

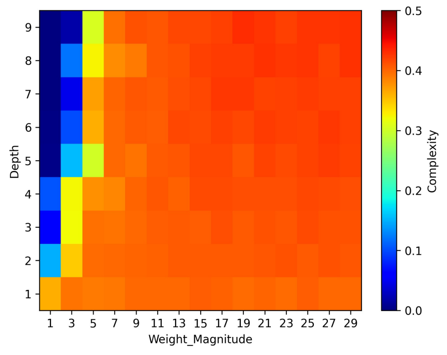


(b) Depth and weights' magnitudes

Figure 4.1: Complexity of ReLU-based networks



(a) Width and weights' magnitudes



(b) Depth and weights' magnitudes

Figure 4.2: Complexity of ReLU-based networks with positional encoding

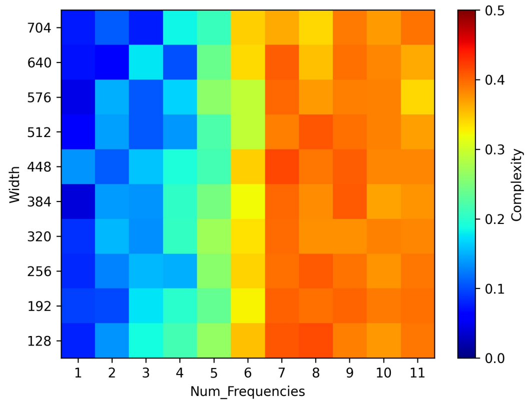
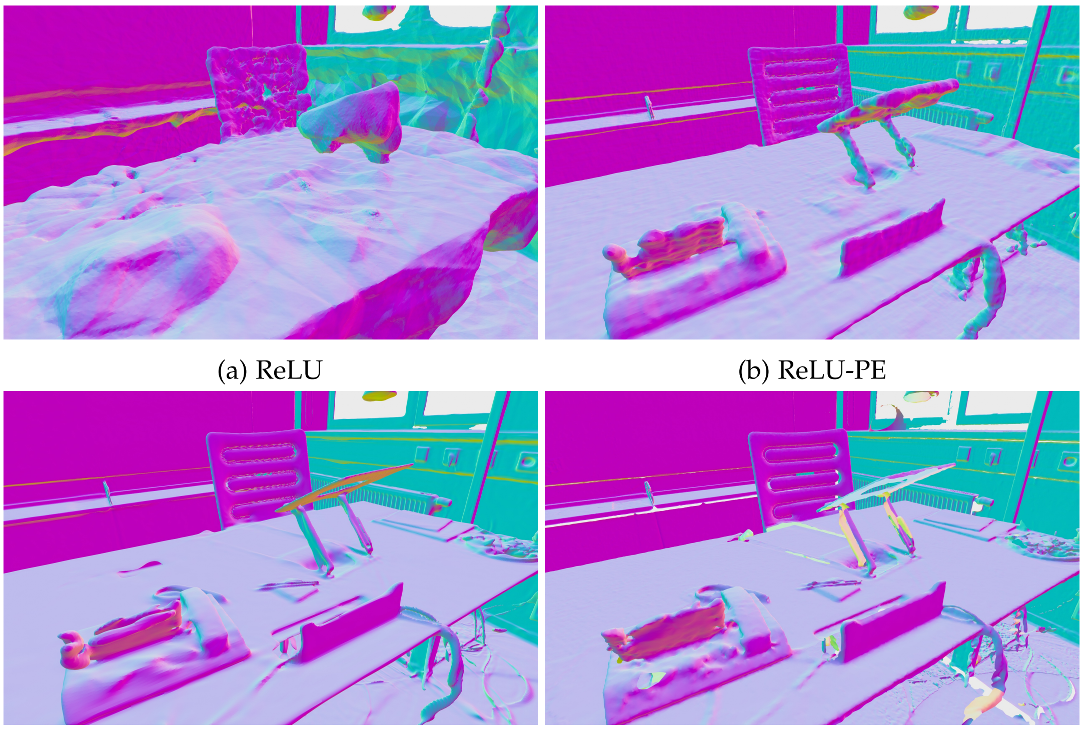


Figure 4.3: Complexity of ReLU-based networks depending on the degree of positional encoding

Positional encoding ($L = 10$) substantially improves the ability of ReLU-based INRs to capture fine-grained detail as indicated by Figure 4.3. Nonetheless, ReLU-based networks still perform poorly when tasked with learning an SDF from a point cloud with normals, an outcome supported by the findings in the SIREN paper [4]. Figure 4.4 shows that a plain ReLU INR cannot model high-frequency structure at all, and ReLU-PE significantly outperforming plain ReLU INR. However, even with positional encoding ReLU-PE underperforms on both high-frequency and low-frequency regions comparing to SIREN reconstruction.

Method	$\downarrow$ **Chamfer Distance,** $\cdot 10^{-3}$	$\uparrow$ **Completeness 0.3%**	$\downarrow$ **LPIPS**
ReLU	4.108	74.83	0.3546
ReLU-PE	<u>2.172</u>	<u>87.00</u>	<u>0.2581</u>
SIREN	**1.878**	**92.17**	**0.1762**

Table 4.1: Metrics for the whole mesh



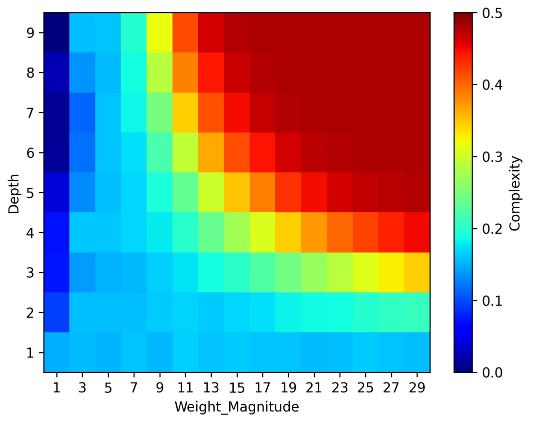
(a) ReLU (b) ReLU-PE

(c) SIREN (d) Ground Truth

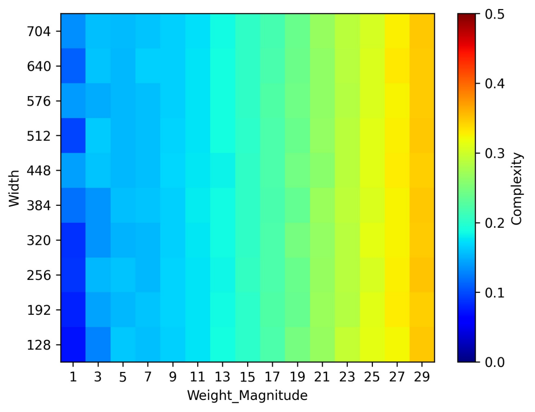
Figure 4.4: Quality of the office room reconstruction

4.1.2 SIREN

We evaluated SIREN in a similar way as ReLU-based networks. Figures 4.5 and 4.6 indicate that network width has a negligible influence on the complexity of the generated signal, whereas depth, weight magnitude, and ω play a significant role.



(a) Depth and Weights' magnitude



(b) Width and Weights' magnitude

Figure 4.5: Depth and magnitudes of initialized weights affect the complexity of SIREN

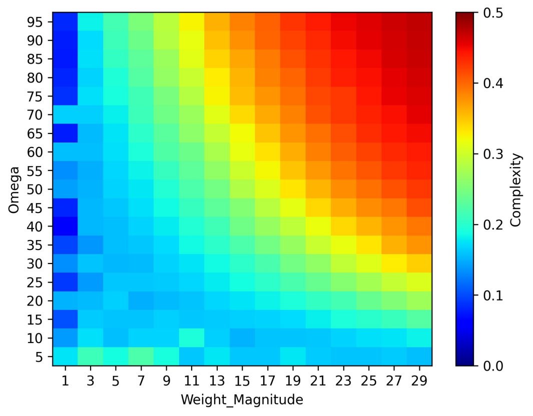


Figure 4.6: ω and weight magnitudes.

The impact of depth and width of the network on the quality of the reconstruction of a trained network is straightforward and can be explained purely by the network's increased capacity. Detailed results across the tested depths and widths are reported in the tables 4.2 and 4.3.

Method	$\downarrow$ **Chamfer Distance,** $\cdot 10^{-3}$	$\uparrow$ **Completeness 0.3%**	$\downarrow$ **LPIPS**
SIREN 3 layer	2.106	87.94	0.2258
SIREN 5 layers	<u>1.878</u>	<u>92.17</u>	**0.1762**
SIREN 7 layers	**1.789**	**93.43**	<u>0.1838</u>

Table 4.2: Metrics for models with a different number of hidden layers

Method	$\downarrow$ **Chamfer Distance,** $\cdot 10^{-3}$	$\uparrow$ **Completeness 0.3%**	$\downarrow$ **LPIPS**
SIREN width = 512	1.881	91.84	<u>0.1801</u>
SIREN width = 1024	<u>1.878</u>	<u>92.17</u>	**0.1762**
SIREN width = 1536	**1.821**	**92.97**	0.1857

Table 4.3: Metrics for models with different hidden layer widths

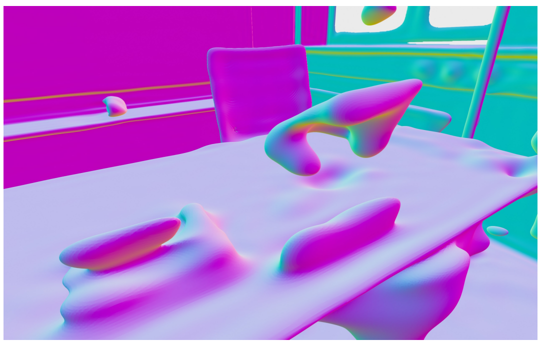
Table 4.4 reveals that the model with $\omega = 30$ delivers the best overall performance, aligning with the default setting adopted in the original SIREN paper. Results for high and low-frequency regions separately are given in the Table 4.5. Separate metrics for high- and low-frequency regions are presented in Table 4.5. Although reconstructions appear overly smooth at $\omega = 5$ and become markedly noisy for $\omega \geq 75$, there is no clear linear relationship between ω and reconstruction quality in either frequency band. Figure 4.7 shows examples of reconstructed meshes.

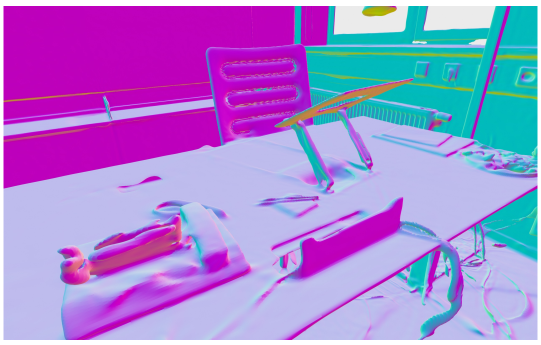
Method	$\downarrow$ **Chamfer Distance,** $\cdot 10^{-3}$	$\uparrow$ **Completeness 0.3%**	$\downarrow$ **LPIPS**
SIREN $\omega = 5$	2.178	86.42	0.2378
SIREN $\omega = 15$	<u>1.904</u>	91.44	0.1855
SIREN $\omega = 30$	**1.878**	<u>92.17</u>	**0.1762**
SIREN $\omega = 45$	1.941	**92.23**	<u>0.1805</u>
SIREN $\omega = 75$	1.944	90.99	0.3458
SIREN $\omega = 100$	2.064	88.93	0.3764

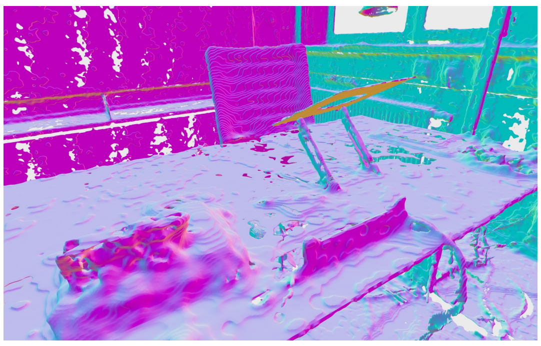
Table 4.4: Metrics for models with different ω

Method	$\downarrow$ **Chamfer Distance, High,** $\cdot 10^{-3}$	$\downarrow$ **Chamfer Distance, Low,** $\cdot 10^{-3}$
SIREN $\omega = 5$	3.173	2.064
SIREN $\omega = 15$	2.309	<u>2.009</u>
SIREN $\omega = 30$	2.210	**2.002**
SIREN $\omega = 45$	**2.123**	2.025
SIREN $\omega = 75$	**2.123**	2.193
SIREN $\omega = 100$	2.305	2.236

Table 4.5: Metrics for models with different ω: high and low-frequency regions


(a) $\omega = 5$


(b) $\omega = 30$


(c) $\omega = 75$

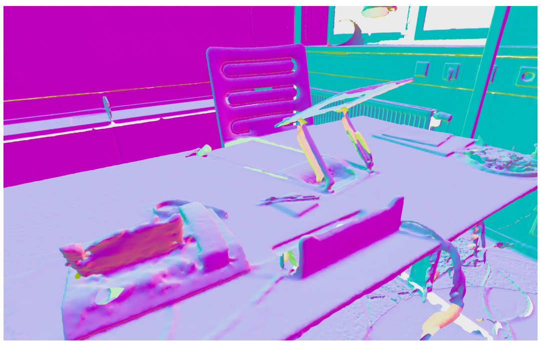

(d) Ground Truth

Figure 4.7: Quality of the office room reconstruction, SIREN for different ω values

4.2 Evaluating SIREN-FM

After the experiments showed that frequency scaling coefficients ω greatly affect the performance of the SIREN-based implicit neural representations, we proceeded to evaluate our proposed architecture in which every layer's sine activation is modulated by a trainable coefficient predicted from the input coordinates (see Section 3.4).

We benchmark SIREN-FM against other implicit neural representation models. As baselines, we include the original SIREN [4], upon which our approach is based, and FINER [5], a popular SIREN extension that incorporates implicit frequency modulation. To ensure comparable model capacities, we adjust hidden-layer widths so that all networks have roughly the same number of parameters: the number of neurons in hidden layers is 256 for SIREN-FM (accounting for its trainable modulator), and it is 300 for both SIREN and FINER.

Method	**SIREN**	**FINER**	**Ours**
Parameters	273k	273k	283k

Table 4.6: Number of trainable parameters

ModSiren [13] (section 2.7) failed to converge under our experimental setup, which adopts the original SIREN training scheme and loss formulation. In the ModSiren paper, the only 3D experiment involves direct regression on a dense SDF volume, where the model performs well. Our findings therefore suggest that the failure arises from the greater challenge of learning an SDF from a point cloud with normals rather than from any implementation error, see Supplementary Materials 6.1.

STAF [6] (section 2.6) was also evaluated. Although the network converged, all trials produced results worse than SIREN, so this data is omitted. As with ModSiren, STAF converges on direct SDF volume regression and yields high-quality meshes as shown in the Supplementary Materials 6.2.

4.2.1 Quantitative results

Metrics for the machinery room scene

SIREN-FM outperforms SIREN and FINER regarding Chamfer distance and Completeness score (table 4.7). When evaluated on the high-frequency and low-frequency regions separately (tables 4.8 and 4.9), SIREN-FM again leads in both Chamfer distance and Completeness (note that these classes are not exhaustive, some parts of the scene fall outside either class). In contrast, SIREN achieves the best LPIPS score, seemingly because of the smoother surfaces. As Figure 4.8 shows, the FINER's surfaces are noisier than SIREN's, and SIREN-FM introduces a grid-like pattern more noticeable on the floor and tiles in Figure 4.11.

Method	$\downarrow$ **Chamfer Distance,** $\cdot 10^{-3}$	$\uparrow$ **Completeness 0.2%**	$\downarrow$ **LPIPS**
SIREN	1.390	87.36	**0.2077**
FINER	1.306	88.53	0.2138
Ours	**1.289**	**88.57**	0.2175

Table 4.7: Metrics for the whole mesh

Method	$\downarrow$ **Chamfer Distance, HF,** $\cdot 10^{-3}$	$\uparrow$ **Completeness 0.2%, HF**
SIREN	1.720	78.37
FINER	1.563	80.74
Ours	**1.529**	**81.30**

Table 4.8: Metrics for the high-frequency regions

Method	$\downarrow$ **Chamfer Distance, LF,** $\cdot 10^{-3}$	$\uparrow$ **Completeness 0.2%, LF**
SIREN	1.123	93.17
FINER	1.101	93.38
Ours	**1.045**	**93.66**

Table 4.9: Metrics for the low-frequency regions

Metrics for the office room scene

Across the entire mesh, SIREN-FM achieves a lower Chamfer distance than FINER, while showing worse LPIPS scores (table 4.10). Completeness scores for all models are nearly identical and effectively indistinguishable. SIREN-FM performs best when evaluated in low-frequency regions and places second after FINER in high-frequency regions (tables 4.11 and 4.12).

Method	$\downarrow$ **Chamfer Distance,** $\cdot 10^{-3}$	$\uparrow$ **Completeness 0.3%**	$\downarrow$ **LPIPS**
SIREN	<u>1.776</u>	92.13	<u>0.1929</u>
FINER	1.812	**92.16**	**0.189**
Ours	**1.744**	<u>92.14</u>	0.2221

Table 4.10: Metrics for the whole mesh

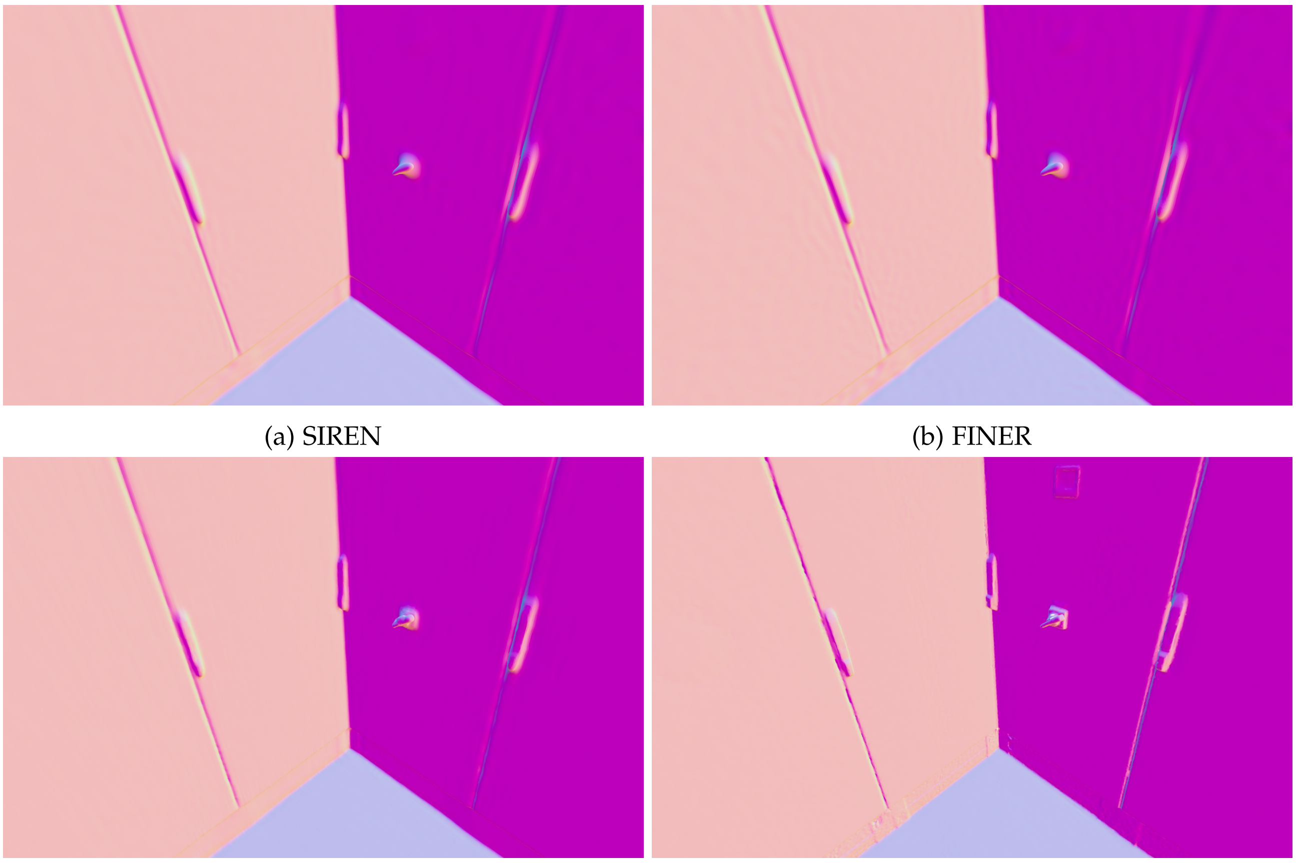
Method	$\downarrow$ **Chamfer Distance, HR,** $\cdot 10^{-3}$	$\uparrow$ **Completeness 0.3%, HR**
SIREN	2.441	82.62
FINER	**2.164**	**84.13**
Ours	<u>2.270</u>	<u>83.23</u>

Table 4.11: Metrics for the high-frequency regions

Method	$\downarrow$ **Chamfer Distance, LF,** $\cdot 10^{-3}$	$\uparrow$ **Completeness 0.3%, LF**
SIREN	<u>1.590</u>	<u>98.01</u>
FINER	1.663	97.90
Ours	**1.566**	**98.13**

Table 4.12: Metrics for low-frequency regions

4.2.2 Qualitative results



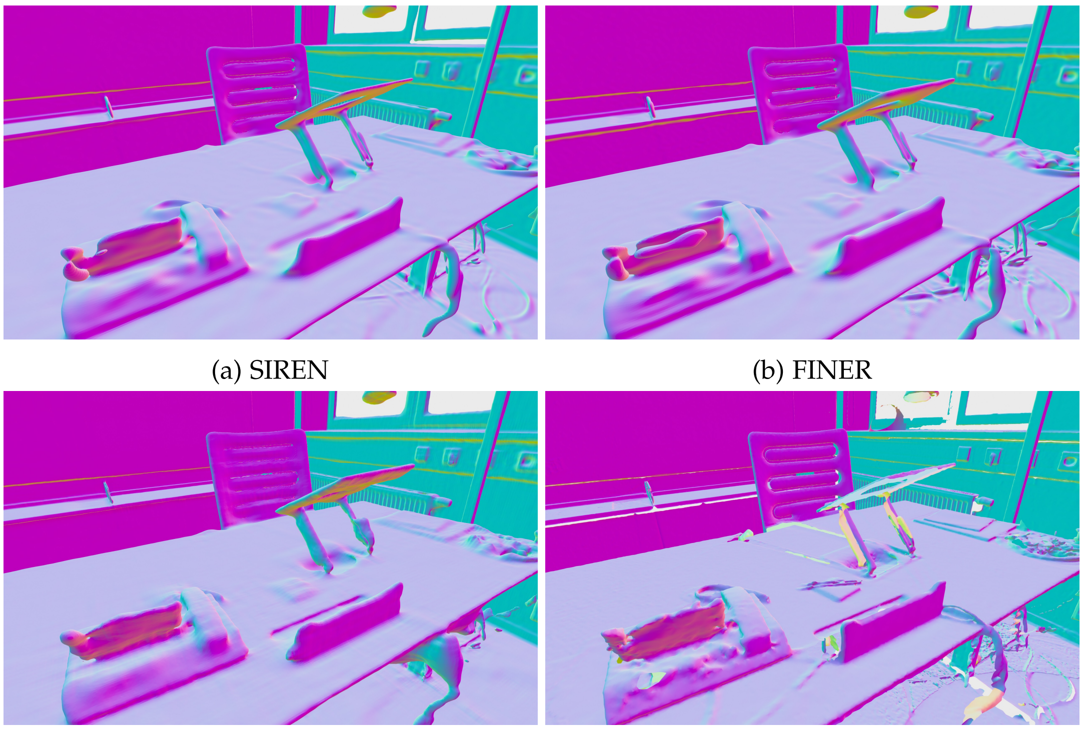
(a) SIREN

(b) FINER

(c) SIREN-FM

(d) Ground Truth

Figure 4.8: Quality of smooth surface reconstruction



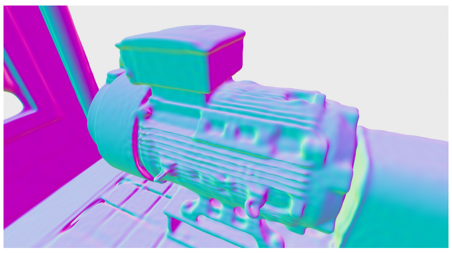
(a) SIREN

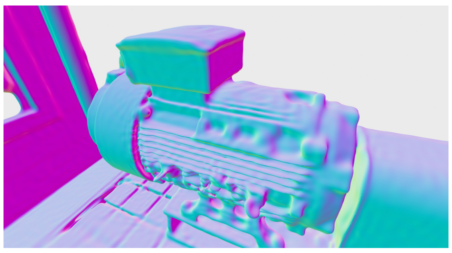
(b) FINER

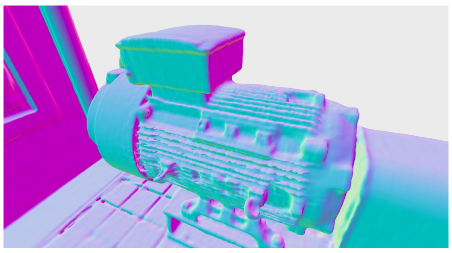
(c) SIREN-FM

(d) Ground Truth

Figure 4.9: Quality of the office room reconstruction


(a) SIREN


(b) FINER


(c) SIREN-FM

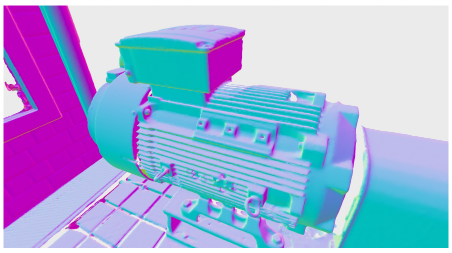
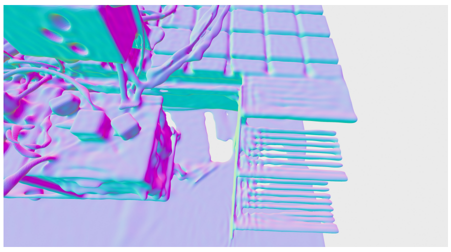
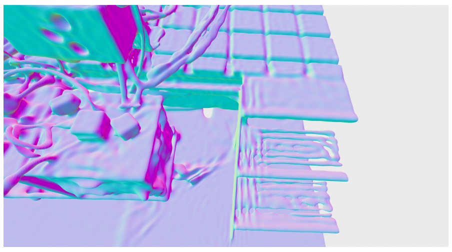
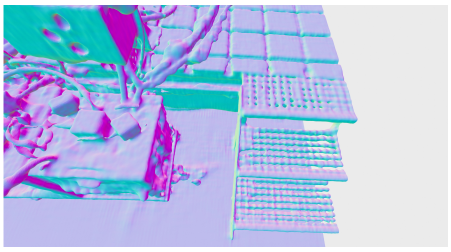

(d) Ground Truth

Figure 4.10: Quality of high-frequency regions reconstruction: compressor


(a) SIREN


(b) FINER


(c) SIREN-FM

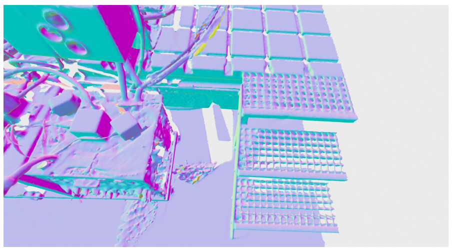

(d) Ground Truth

Figure 4.11: Quality of high-frequency regions reconstruction: stairs

4.3 Robustness of the SIREN-FM to different choices of the base ω

Sensitivity of SIREN to the choice of the hyperparameter ω is one of the downfalls of the model. Tables 4.13 and 4.14 show that SIREN-FM is substantially more robust to changes in this setting. Interestingly, the mean values of β near the mesh surface vary roughly inversely with the base frequency ω. For example, for $\omega = 10$: $[\beta_1, \beta_2, \beta_3, \beta_4] = [1.17, 0.87, 0.95, 1.22]$ and for $\omega = 20$ values drop to $[\beta_1, \beta_2, \beta_3, \beta_4] = [1.03, 0.72, 0.72, 0.81]$. Although the coefficients at $\omega = 10$ are not precisely double those at $\omega = 20$, this can be explained by the contributions of other learned parameters, specifically the amplitudes α_i and phase shifts γ_i, which also modulate the overall effect.

Method	$\downarrow$ **Chamfer Distance,** $\cdot 10^{-3}$	$\downarrow$ **Mean**	$\downarrow$ **Std**
SIREN $\omega = 10$	1.451		
SIREN $\omega = 20$	1.358	1.399	0.038
SIREN $\omega = 30$	1.390		
FINER $\omega = 10$	1.515		
FINER $\omega = 20$	1.469	1.43	0.089
FINER $\omega = 30$	1.306		
Ours $\omega = 10$	1.294		
Ours $\omega = 20$	1.347	**1.320**	**0.021**
Ours $\omega = 30$	1.321		

Table 4.13: Robustness of SIREN-FM comparing to SIREN and FINER: Chamfer distance

Method	$\uparrow$ Completeness 0.2%	$\uparrow$ Mean	$\downarrow$ Std
SIREN $\omega = 10$	84.90		
SIREN $\omega = 20$	87.14	86.46	1.11
SIREN $\omega = 30$	87.36		
FINER $\omega = 10$	82.84		
FINER $\omega = 20$	83.23	84.86	2.59
FINER $\omega = 30$	88.53		
Ours $\omega = 10$	88.41		
Ours $\omega = 20$	88.34	**88.51**	**0.20**
Ours $\omega = 30$	88.80		

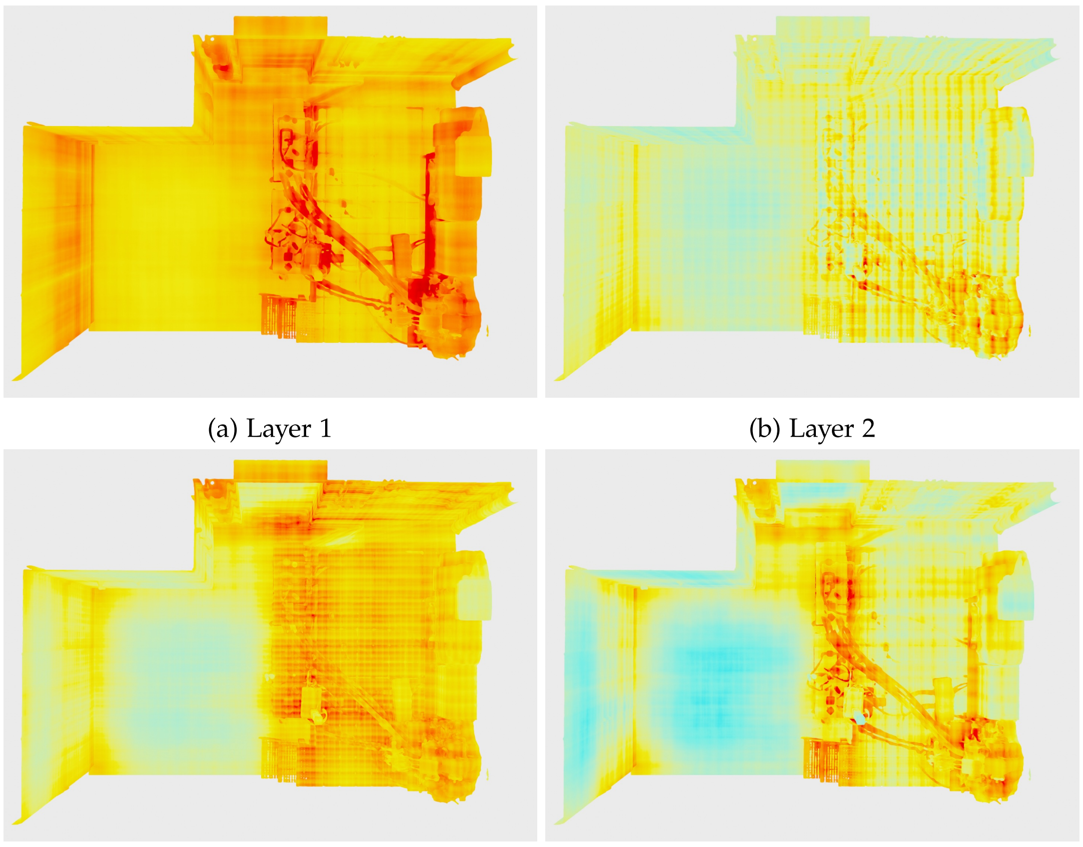
Table 4.14: Robustness of SIREN-FM comparing to SIREN and FINER: Completeness

4.4 Visualizing the predicted coefficients of SIREN-FM

The modulator predicts an amplitude, frequency, and phase shift of the hidden layers' activations.

$$\sigma_i(\cdot) = \alpha_i \cdot \sin(\beta_i \omega x + \gamma_i)$$

Here, $\beta_i \omega$ acts as the frequency scaling factor. To inspect its spatial distribution, we colour the reconstructed surface by the β_i value of the closest point on a dense $512 \times 512 \times 512$ grid spanning $[-1, 1]^3$ (see Figure 4.12). Regions containing fine-grained detail display consistently higher β_i across all layers, with the effect being most pronounced in the first and fourth layers.



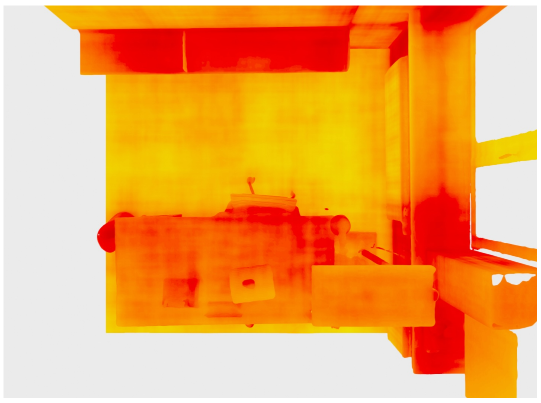
(a) Layer 1

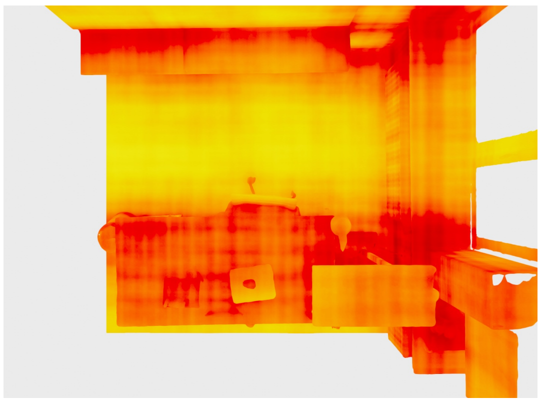
(b) Layer 2

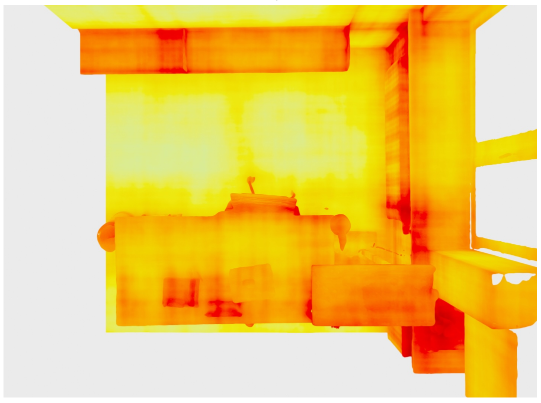
(c) Layer 3

(d) Layer 4

Figure 4.12: β_i values heatmap, red is the maximum β_i, machinery room


(a) Layer 1


(b) Layer 2


(c) Layer 3

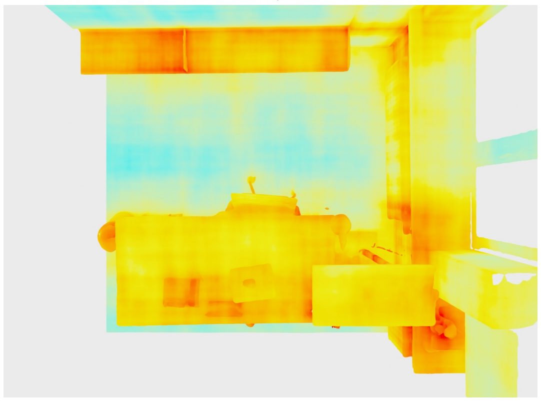

(d) Layer 4

Figure 4.13: β_i values heatmap, red is the maximum β_i, office room

5 Discussion

5.1 Inductive biases of various architectures

5.1.1 ReLU and ReLU + PE

Results for the complexity of the randomly initialized and untrained ReLU-based 3D implicit neural representations align with the findings of the authors of the Neural Redshift paper about 2D outputs of ReLU-based networks. Introducing positional encoding markedly boosts output complexity, with complexity getting higher with the length of the positional encoding. Positional encoding is already a known method to fight against spectral bias, the process when networks prefer to learn low frequency functions [16].

When trained to model an SDF of a 3D scene, a plain ReLU network fails to capture high-frequency details (Figure 4.4), making it unsuitable for this task. Incorporating positional encoding (ReLU-PE) improves reconstruction quality, but the resulting surfaces remain substantially noisier compared to ground truth or surfaces produced by SIREN.

5.1.2 SIREN

Our findings here match the results of the Neural Redshift paper for the networks that use sine as the nonlinearity: increasing depth or the initial weight magnitude boosts the output's Fourier complexity, whereas width has little effect. For trained networks, both depth and width (tables 4.2 and 4.3) improve the Chamfer distance and Completeness scores. This can be explained by network's increased capacity.

Higher values for ω that tune the activation functions in the SIREN layer show higher Fourier complexity in untrained SIREN, but there is no direct correlation between the network's ω and performance on high and low-frequency regions (tables 4.4 and 4.5). Instead, there is some range for ω values that perform better on the particular sample, as described in the SIREN paper and observed in our experiments (table 4.4).

Intuitively, the complexity of the outputs of the untrained neural networks does not define how well the network can encode high-frequency or low-frequency information. The complexity of the random networks hints at which degrees of complexity the network can achieve, i.e., it can show the level that the network can achieve, but cannot guarantee that the trained network performs better on the real task. Still, numerous works have chosen SIREN's activations as a possible target for improvement. The authors of FINER reported that using an adaptive ω can improve the performance for high-frequency tasks and relieve the SIREN's spectral bias. The authors of STAF proposed to make the amplitude, frequency scale, and phase shift trainable to allow even larger degrees of freedom. We propose to predict these coefficients dynamically, depending on the input coordinate, thus enabling the activations to change for different regions of our input space.

5.2 Evaluating SIREN-FM

5.2.1 3D scene reconstruction

SIREN-FM shows good results on both 3D scenes used in this work: the office room ($\sim$ 4.5 million points) and the machinery room ($\sim$ 9.1 million points). In the machinery scene, it overperforms both SIREN and FINER in Chamfer distance and Completeness. Still, these geometric gains come with a trade-off: LPIPS scores for SIREN-FM are worse than both baselines. SIREN-FM surfaces are noisier than SIREN's and have a grid-like pattern more noticeable on the floor and tiles in Figure 4.11. By contrast, SIREN's smoother surfaces achieve the best LPIPS scores.

In terms of geometry, SIREN-FM is capable of reconstructing fine details like small holes in the stairs on the Figure 4.11, where our architecture is doing significantly better than the baselines.

In the office room scene, SIREN-FM again outperforms SIREN and FINER regarding Chamfer distance, while all models are tied in completeness score. Also, SIREN-FM shows better results in low-frequency regions but concedes to FINER when evaluated in high-frequency regions. This might be explained by the fact that the office room has fewer chaotic high-frequency regions, like a compressor with wires in the machinery room scene, and SIREN-FM cannot realize its full potential.

These improvements were achieved without significant parameter overhead, as the width of the hidden size for SIREN-FM was reduced comparing to baselines in order to keep the number of parameters in the same range.

5.2.2 Robustness of the SIREN-FM depending on ω

The dependence of SIREN architecture on the hyperparameter ω is a primary drawback that extensions such as FINER, STAF, and our model try to solve. As shown in 4.13 and 4.14, SIREN-FM is markedly more robust to the predefined ω values and finds the adaptive coefficients to compensate for different initial ω. This behaviour suggests that there is an adaptation of the network in an attempt to better match the required frequency and the data.

5.2.3 Analyzing predicted coefficients of the SIREN-FM

The β_i coefficients visualized on the mesh (figure 4.12) are consistently larger in regions containing high-frequency detail, which supports our idea about learning parameters of the activation being useful for the model. For some cases, it is not clear why particular regions have higher predicted β values, but the distribution of these coefficients has a concrete spatial dependency and is not purely random.

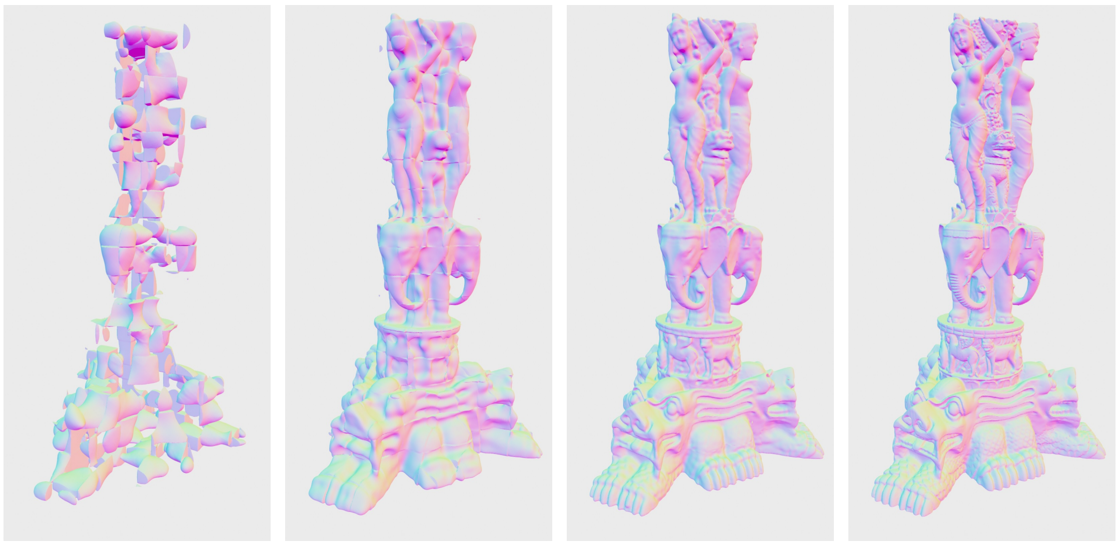
5.3 Conclusion and future directions

SIREN-FM suggests a novel view on the architecture of implicit neural representations and proposes activation functions directly tuned by the input coordinate. Our results prove that SIREN-FM consistently outperforms SIREN on geometrical metrics and successfully competes in performance with FINER.

One of the future directions would be to explore the performance of SIREN-FM on other scenes and possibly bigger models. One of the apparent drawbacks of our architecture is grid-like artifacts on the reconstructed surfaces. Experimenting on the modulator architecture, for example, training with smaller positional encoding (and limiting the innate complexity of the network), could alleviate this problem.

6 Supplementary materials

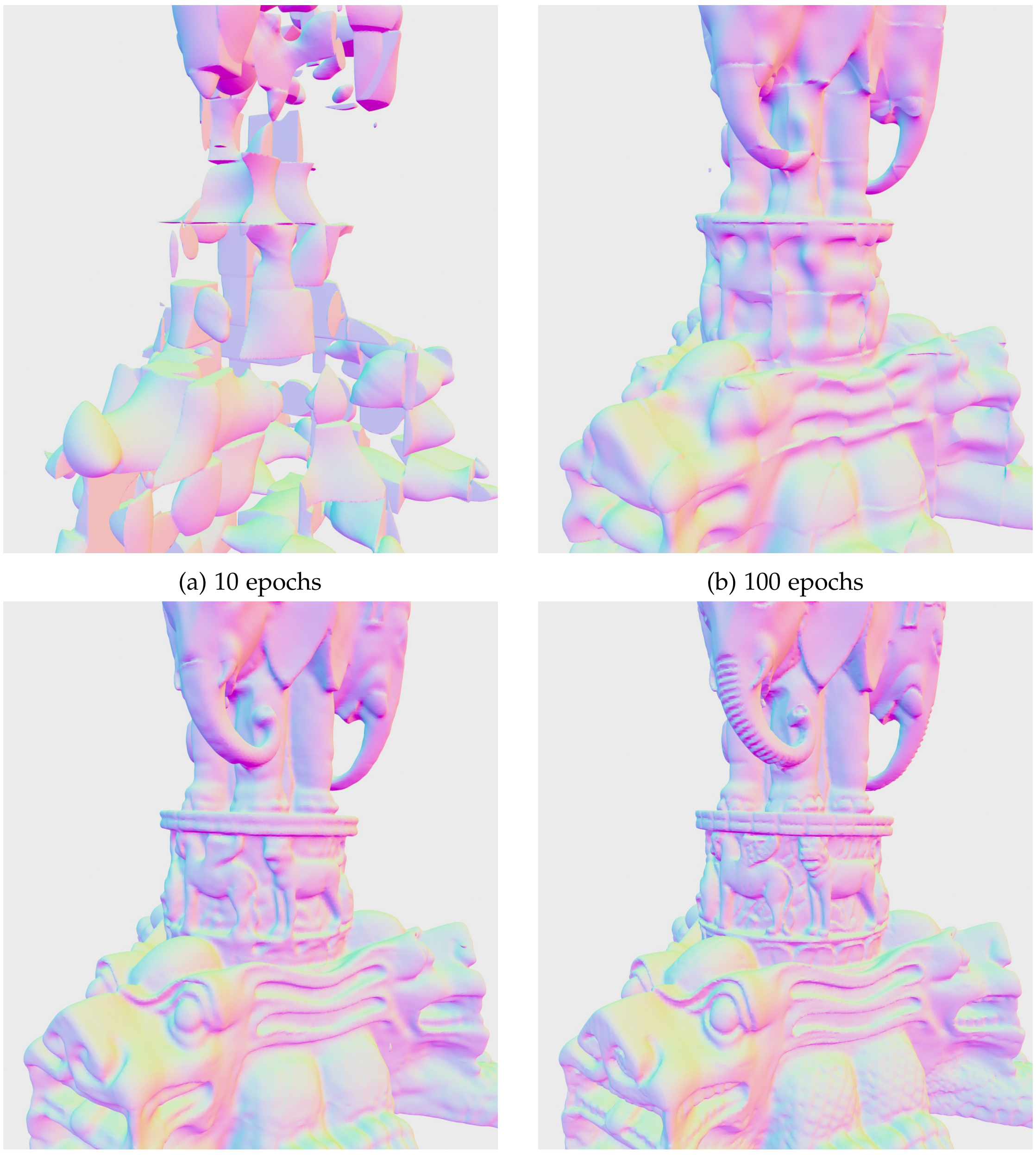
6.1 ModSiren



(a) 10 epochs (b) 100 epochs (c) 3000 epochs (d) Ground Truth

Figure 6.1: ModSiren training on SDF volume

Training with the direct regression of the SDF volume converges well. In Figure 6.1 for 10 and 100 epochs, it is possible to see the contours of cubes, 3D "tiles" from the paper, that gradually converge to the correct and smooth shape. Same for the close-up Figure 6.2.



(a) 10 epochs

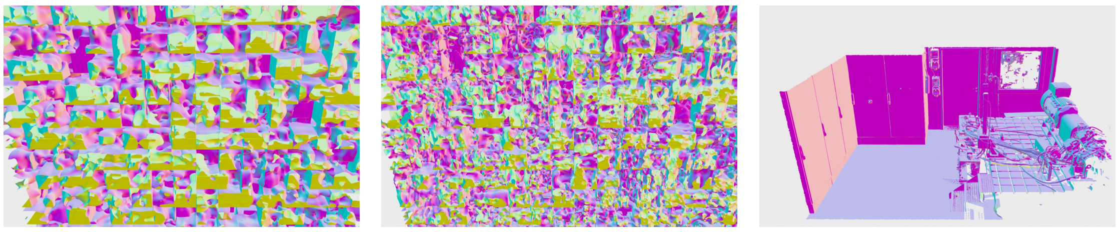
(b) 100 epochs

(c) 3000 epochs

(d) Ground Truth

Figure 6.2: ModSiren training on SDF volume: close-up

Training with the method adopted from SIREN, based on points with normals, on the Figure 6.3, the model's predictions do not significantly change from the initial cubic contours of 3D tiles, and there is no significant progress across the epochs in this type of training.

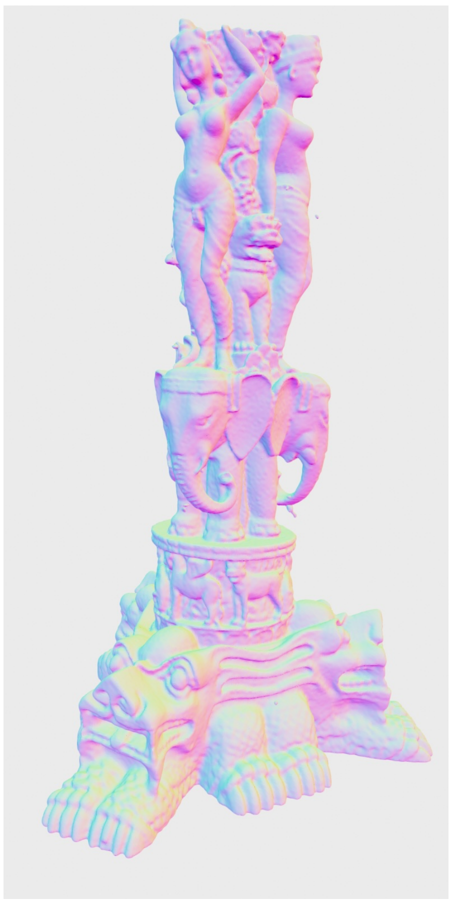


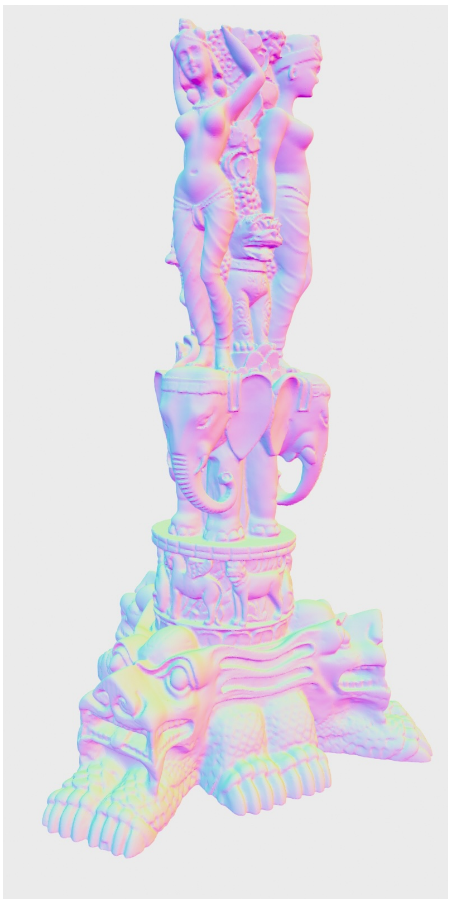
(a) 25 epochs (b) 1000 epochs (c) Ground Truth

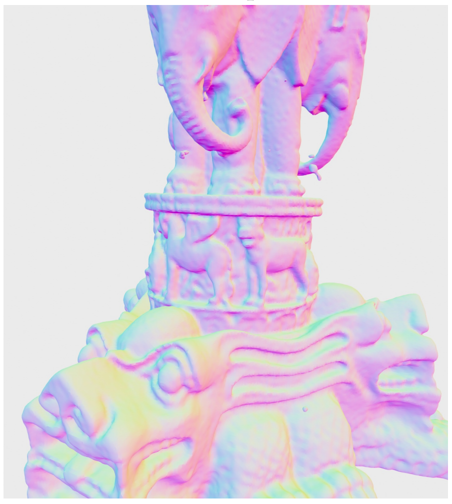
Figure 6.3: ModSiren training on point clouds with normals

6.2 STAF

STAF converges on the direct SDF volume regression task and accurately reconstructs the signal.


(a) 3000 epochs


(b) Ground Truth


(c) 3000 epochs

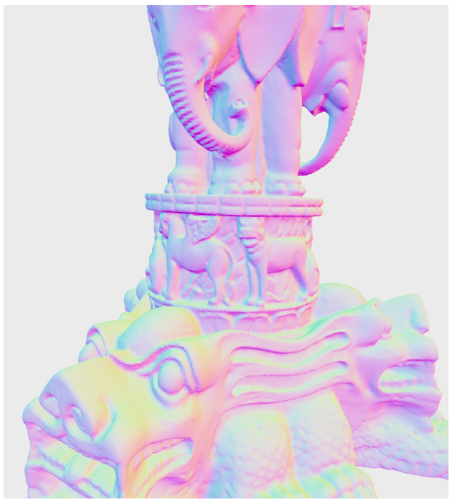
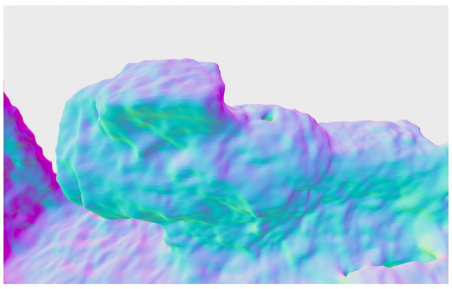
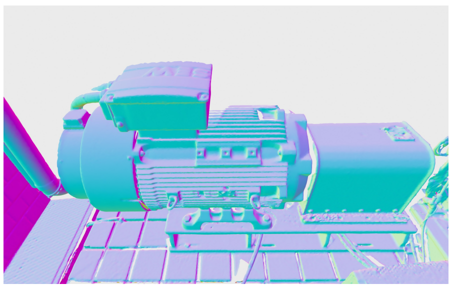
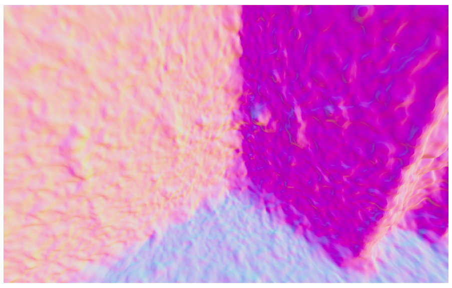

(d) Ground Truth

Figure 6.4: STAF training on SDF volume

As a 3D shape reconstruction experiment, the authors only attempted training the network to predict the dense occupancy volume. During the SIREN-style training, STAF is unable to learn the details of the more complex 3D scene.


(a) 1000 epochs


(b) Ground Truth


(c) 1000 epochs

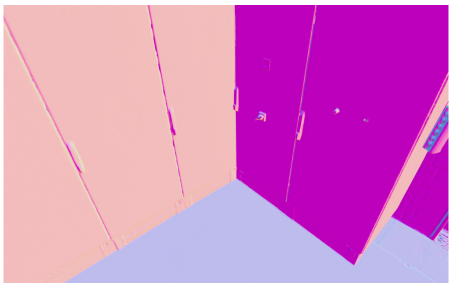

(d) Ground Truth

Figure 6.5: STAF training on point clouds with normals

List of Figures

List of Tables

Bibliography

[1] D. Teney, A. Nicolicioiu, V. Hartmann, and E. Abbasnejad. "Neural Redshift: Random Networks are not Random Functions." In: *CVPR*. 2024.

[2] D. Teney, L. Jiang, F. Gogianu, and E. Abbasnejad. "Do We Always Need the Simplicity Bias? Looking for Optimal Inductive Biases in the Wild." In: *CVPR*. 2025.

[3] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. A. Hamprecht, Y. Bengio, and A. Courville. "On the Spectral Bias of Neural Networks." In: *ICML*. 2019.

[4] V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein. "Implicit Neural Representations with Periodic Activation Functions." In: *NeurIPS*. 2020.

[5] Z. Liu, H. Zhu, Q. Zhang, J. Fu, W. Deng, Z. Ma, Y. Guo, and X. Cao. "FINER: Flexible spectral-bias tuning in Implicit NEural Representation by Variable-periodic Activation Functions." In: *CVPR*. 2024.

[6] A. Morsali, M. Vaez, M. H. Soltani, A. Kazerouni, and M. Mohammad-Noori. "STAF: Sinusoidal Trainable Activation Functions for Implicit Neural Representation." In: *arXiv preprint* (2025).

[7] J. C. Lee, D. Rho, S. Nam, J. H. Ko, and E. Park. "Coordinate-Aware Modulation for Neural Fields." In: *arXiv preprint* (2023).

[8] J. N. P. Martel, D. B. Lindell, C. Z. Lin, E. R. Chan, M. Monteiro, and G. Wetzstein. "ACORN: Adaptive Coordinate Networks for Neural Scene Representation." In: *SIGGRAPH*. 2021.

[9] D. B. Lindell, D. V. Veen, J. J. Park, and G. Wetzstein. "BACON: Band-limited Coordinate Networks for Multiscale Scene Representation." In: *CVPR*. 2022.

[10] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng. "Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains." In: *NeurIPS*. 2020.

[11] H. Zhu, Z. Liu, Q. Zhang, J. Fu, W. Deng, Z. Ma, Y. Guo, and X. Cao. “FINER++: Building a Family of Variable-periodic Functions for Activating Implicit Neural Representation.” In: *arXiv preprint* (2024).

[12] A. Kazerouni, R. Azad, A. Hosseini, D. Merhof, and U. Bagci. “INCODE: Implicit Neural Conditioning with Prior Knowledge Embeddings.” In: *arXiv preprint* (2023).

[13] I. Mehta, M. Gharbi, C. Barnes, E. Shechtman, R. Ramamoorthi, and M. Chandraker. “Modulated Periodic Activations for Generalizable Local Functional Representations.” In: *ICCV*. 2021.

[14] R. Fathony, A. K. Sahu, D. Willmott, and J. Z. Kolter. “Multiplicative Filter Networks.” In: *ICLR*. 2021.

[15] C. Yeshwanth, Y.-C. Liu, M. Nießner, and A. Dai. “ScanNet++: A High-Fidelity Dataset of 3D Indoor Scenes.” In: *ICCV*. 2023.

[16] 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.” In: *ECCV*. 2020.