MINER: Multiscale Implicit Neural Representation

Vishwanath Saragadam(✉), Jasper Tan, Guha Balakrishnan, Richard G. Baraniuk, and Ashok Veeraraghavan

Rice University, Houston, TX 77005, USA
vishwanath.saragadam@rice.edu

Abstract. We introduce a new neural signal model designed for efficient high-resolution representation of large-scale signals. The key innovation in our *multiscale implicit neural representation* (MINER) is an internal representation via a Laplacian pyramid, which provides a sparse multiscale decomposition of the signal that captures orthogonal parts of the signal across scales. We leverage the advantages of the Laplacian pyramid by representing small disjoint patches of the pyramid at each scale with a small MLP. This enables the capacity of the network to adaptively increase from coarse to fine scales, and only represent parts of the signal with strong signal energy. The parameters of each MLP are optimized from coarse-to-fine scale which results in faster approximations at coarser scales, thereby ultimately an extremely fast training process. We apply MINER to a range of large-scale signal representation tasks, including gigapixel images and very large point clouds, and demonstrate that it requires fewer than 25% of the parameters, 33% of the memory footprint, and 10% of the computation time of competing techniques such as ACORN to reach the same representation accuracy. A fast implementation of MINER for images and 3D volumes is accessible from https://vishwa91.github.io/miner.

1 Introduction

Neural implicit representations have emerged as a promising paradigm for signal representation and interpolation with pervasive applications in 3D view synthesis [9,16,19,23], images [3], video [2], and linear inverse problems [3,25]. At the core of such neural representations is one or several multi layer perceptrons (MLPs) that produce a continuous mapping from signal coordinates to the values of the signal at those coordinates.

The success of neural implicit representations relies on the ability to fit models accurately (high representation accuracy), rapidly (short training time), and in a concise manner (small number of parameters). However, most state-of-the-art implicit representations require training a single large MLP (parameters in

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-20050-2_19.

S. Avidan et al. (Eds.): ECCV 2022, LNCS 13683, pp. 318–333, 2022.
https://doi.org/10.1007/978-3-031-20050-2_19

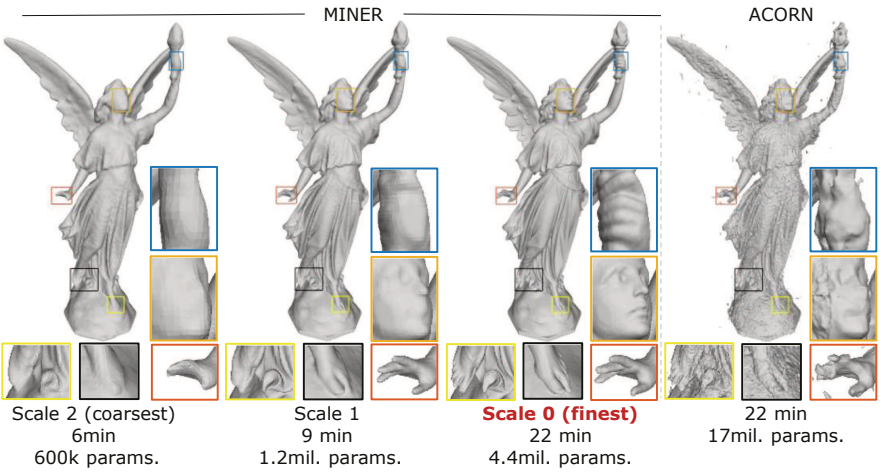


Fig. 1. Multiscale Implicit Representations. We present a novel implicit representation framework called MINER for large visual signals such as images, videos, and 3D volumes. We leverage the self-similarity of visual signals across scales to iteratively represent models from coarse to fine scales, resulting in a dramatic decrease in inference and training time, while requiring fewer parameters and less memory than state-of-the-art representations. This figure demonstrates fitting of the Lucy 3D mesh over three scales with scale 2 being the coarsest and 0 being the finest. MINER achieves high quality results across all scales with high IoU value and achieves an IoU of 0.999 at the finest scale in less than 30 min. In comparison, the state-of-the-art approach (ACORN) results in an IoU of 0.97 in that time, while requiring far more parameters.

millions) that suffers from high computational cost, requiring large memory footprints and long training times. While there have been several modifications to the network architecture [13,20,22] and inference [31], neural implicit representations are not yet practical for handling extremely high dimensional signals such as gigapixel images or 3D point clouds with several billion data points.

We introduce a multiscale implicit neural representation (MINER) that is well-suited for representing very high dimensional signals in a concise manner. Our key observation is that Laplacian pyramids of visual signals offer a sparse and multiscale decomposition that naturally separates a signal's frequency content across spatial scales. We leverage the multiscale decomposition by representing each spatial scale of the Laplacian pyramid with different MLPs. Instead of using a single MLP at each scale, we represent a small disjoint image/volume patch of fixed size with a small MLP, resulting in both a multiscale and multipatch decomposition. Such a multipatch decomposition is well-suited for sparse signals as most patches will have near-zero intensity, thereby not requiring an explicit MLP for that patch. MINER enables a fast and flexible multi-resolution analysis, as representing the signal at lower resolution requires training 2× fewer MLPs along each spatial dimension (due to fewer patches), An example on fitting a 3D volume across three spatial scales on one billion points is shown in

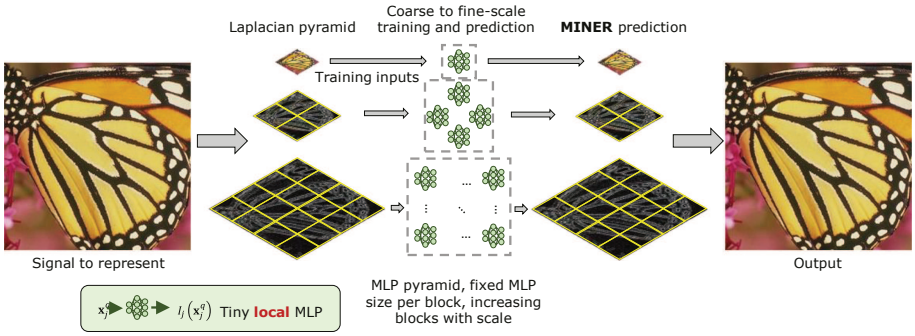


Fig. 2. MINER trains and predicts Laplacian pyramids. Visual signals are similar across scales and are compactly represented by Laplacian pyramids. MINER follows a similar scheme where each scale of the Laplacian pyramid is represented by multiple, local MLPs with a small number of parameters. The number of such MLPs increase by a factor of 2 from coarse to fine scale, thereby representing a fixed spatial size at each scale. This multi-scale representation naturally lends itself to a sequential, coarse-to-fine scale training process that is fast and memory efficient.

Fig. 1. MINER provides a visually pleasing result even at the coarsest spatial scale in six minutes with as few as 600k parameters. The finest scale converges in 22 min. In contrast, for the same amount of training time, state-of-the-art approaches such as ACORN result in many artifacts while also requiring 4× more parameters. An overview of the MINER signal model is shown in Fig. 2.

The multiscale, multi-MLP architecture lends itself to a fast and memory efficient training procedure. At each spatial scale, the parameters of the MLPs are trained for the corresponding Laplacian pyramid scale. We then *sequentially* train MLPs from coarsest scale to the finest scale. The near-orthogonality of the Laplacian transform across scales ensures that new information is added at every scale, thereby resulting in an iterative refinement framework. We leverage the sparsity of the Laplacian transform by comparing the upsampled signal from the fine block and the target signal at that fine block – if the error in representation (or variance of signal) is smaller than a threshold, we prune out the blocks before training starts. This leads to fewer blocks to train at finer resolutions.

MINER is 10× or more faster compared to state-of-the-art implicit representations in terms of training process for a comparable number of parameters and target accuracy. MINER can represent gigapixel images with greater than 38 dB accuracy in less than three hours, compared to more than a day with techniques such as ACORN [13]. For 3D point clouds, MINER achieves an intersection over union (IoU) of 0.999 or higher in less than three minutes, resulting in two orders of magnitude speed up over ACORN. Due to the multiscale representation, MINER can be used for streaming reconstruction of images, as with JPEG2000 [21], or efficiently sampling for rendering purposes with octrees [31] – making neural representations ready for extremely large scale visual signals.

2 Prior Work

MINER draws inspiration from classical multiscale techniques and more recent neural representations. We outline some of the salient works here to set context.

Implicit Neural Representations. Implicit neural representations learn a continuous mapping from local coordinates to the signal value such as intensity for images and videos, and occupancy value for 3D volumes. The learned models are then used for a myriad of tasks including image representations [3], multi-view rendering [16], and linear inverse problems solving [25]. Recent advances in the choice of coordinate representation [27] and non-linearity [22] have resulted in training processes that have high fitting accuracy. Salient works related to implicit representations include the NeRF representations [16] and its many derivatives that seek to learn the 3D geometry from a set of multi-view images. Despite the interest and success of these implicit representations, current approaches often require disproportionately large number of parameters compared to the signal dimension. This culminates in a large memory footprint and training times, precluding representation of very high-dimensional signals.

Architectural Changes for Faster Learning. Several interesting modifications have been proposed to increase training or inference speed. KiloNeRF [20] and deep local shapes [1] replaced the large MLP with multiple small MLPS that fit only a small, disjoint part of the 3D space. Such approaches dramatically speed up the inference time (often by 60×) and in some cases enable better generalization [15], but they have little to no effect on the training process itself. ACORN [13] utilized an adaptive coordinate decomposition to efficiently fit various signals. By utilizing a combination of integer programming and interpolation, ACORN reduced training time for fitting of images and 3D point clouds by one to two orders of magnitude compared to techniques like SIREN [22] and the convolutional occupancy network [19]. However, ACORN does not leverage the cross-scale similarity of visual signals, and this often leads to long convergence times for very large signals. Moreover, the adaptive optimized blocks requires several hundreds of thousands of gradient steps which can be prohibitively expensive.

Multi-scale Representations. Visual signals are similar across scales, and this has been exploited for a wide variety of applications. In computer vision and image processing, the wavelet transform and Laplacian pyramids are often used to efficiently perform tasks such as image registration [28], optical flow computation [29], and feature extraction [11]. Multi-scale representations such as octrees [4,14] and mip-mapping are used to speed up the rendering pipeline. This has also inspired neural mipmapping techniques [9] that utilize neural networks to represent texture at each scale. Along the same lines, spatially adaptive progressive encoding [6] enables a coarse-to-fine training approach that gradually learns higher spatial frequencies. Multi-scale representations are also utilized for

several linear inverse problems such as multi-scale dictionary learning for denoising [24], compressive sensing [17], and sparse approximation [12]. Some recent works have focused on a level-of-detail approach to neural representations [26] (NGLOD) where the multiple scales are *jointly* learned. The implicit displacement fields (IDF) approach [30] similarly learns a smooth approximation of the surface, along with a high frequency displacement at each spatial point to represent the shape. While efficient in rendering, NGLOD and IDF have no advantage in the training phase, as it relies on training all levels of detail at the same time. MINER also results in an LOD representation, but the underlying approach is significantly different. MINER relies on a block-wise representation at each scale with sequential training from coarse to fine scales, which enables more compact representation with faster training times.

3 MINER

MINER combines Laplacian pyramid with a block decomposition of the signal. We now describe the MINER signal model and the training process.

3.1 Signal Model

Let $\mathbf{x}$ be the coordinate and $\mathrm{I}(\mathbf{x})$ be the target. We will assume that the coordinates lie in $[-1, 1]$. Let $\mathcal{D}_j$ be the domain specific operator that downsamples the signal by j times, and $\mathcal{U}_j$ be the domain-specific operator that upsamples the signal by j times. We will leverage J implicit representations, $I_j(\mathbf{x}) \approx N_j$ for $j \in [0, J-1]$, where N_j is the MLP at the j^{th} level of a Laplacian pyramid, a multiscale representation which separates the input signal into scales capturing unique spatial frequency bands. Two desirable properties of such a bandpass pyramid is that signals across scales are approximately orthogonal to one another [5] and are sparse. We found in our experiments that these properties dramatically reduce the training and inference times compared to a lowpass pyramid such as the Gaussian pyramid (see Fig. 3a).

Letting R_j denote the MLP modeling the residual signal at scale j, our Laplacian pyramid representation may be written as:

$$I_{J-1}(\mathbf{x}) = \mathcal{D}_{J-1}(I)(\mathbf{x}) \approx N_{J-1}(\mathbf{x}) \tag{1}$$

$$I_{J-2}(\mathbf{x}) = \mathcal{D}_{J-2}(I)(\mathbf{x}) \approx R_{J-2}(\mathbf{x}) + \mathcal{U}_2(N_{J-1}(\mathbf{x}/2)) \tag{2}$$

$$\vdots$$

$$I(\mathbf{x}) \approx R_0(\mathbf{x}) + \mathcal{U}_2(N_1(\mathbf{x}/2)) \tag{3}$$

$$\approx N_0(\mathbf{x}) + \mathcal{U}_2(N_1(\mathbf{x}/2)) + \cdots + \mathcal{U}_{2^{J-1}}(N_{J-1}(\mathbf{x}/2^{J-1})), \tag{4}$$

where Eq. (1) is the coarsest representation of the signal. At finer scales (as in Eq. (2)), we write the signal to be approximated as a sum of the upsampled version of the previous scale and a residual term. This results in a recursive multi-resolution representation that naturally shares information across scales.

We make two observations about MINER:

- Signals at coarser resolutions are low-dimensional, and therefore require smaller MLPs. These MLPs are faster for inference, which is beneficial for tasks such as mipmapping and LOD-based rendering.
- The parameters of the MLPs up to scale j only rely on the signal at scales $q = j, j+1, \cdots, J-1$. This implies the MLPs can be trained *sequentially* from coarsest to finest scale. We will see next that this offers a dramatic reduction in training time without sacrificing quality.

Using Multiple MLPs per Scale. Equation (4) implies that obtaining a value at a spatial point $\mathbf{x}$ requires evaluating a total of J MLPs across scales. Such joint evaluation has no computational benefit compared to a single scale approach like SIREN [22] with comparable number of parameters. Further, the residual signals at finer scales are often low-amplitude, a consequence of visual signals being composed of several smooth areas. To leverage this fact and make inference faster, we split the signal into equal sized blocks at each scale. We create an MLP for each block that requires significantly fewer parameters than a single full MLP at that scale. Moreover, blocks with small residual energy can be represented as a zero signal, and do not even need to be represented with an MLP.

We now combine the Laplacian representation with the multi-MLP approach stated above. Let $\widetilde{\mathbf{x}}$ be a local coordinate at the finest scale in a block with coordinate (m, n), where $m \in 1, 2, \cdots M$ is the number of vertical blocks, and $n \in 1, 2, \cdots, N$ is the number of horizontal blocks. To evaluate the signal at $\mathbf{x}$,

$$I(\mathbf{x}) = I\left(\widetilde{\mathbf{x}} + \left[\frac{mH}{M}, \frac{nW}{N}\right]\right) = \mathcal{R}_0^{(m,n)}\left(\widetilde{\mathbf{x}} + \left[\frac{mH}{M}, \frac{nW}{N}\right]\right) + \cdots$$
$$+ \cdots \mathcal{U}_2\left(\mathcal{N}_1^{(\lfloor m/2 \rfloor, \lfloor n/2 \rfloor)}\left(\widetilde{\mathbf{x}} + \left[\left\lfloor \frac{mH}{2M} \right\rfloor, \left\lfloor \frac{nW}{2N} \right\rfloor\right]\right)\right), \tag{5}$$

where $\lfloor \cdot \rfloor$ is the floor operator, and $\mathcal{N}_j^{(m,n)}$ is the MLP for block at (m, n) and at scale j. With this formulation, we require evaluation of at most J *small* MLPs instead of large MLPs, thereby dramatically reducing inference time.

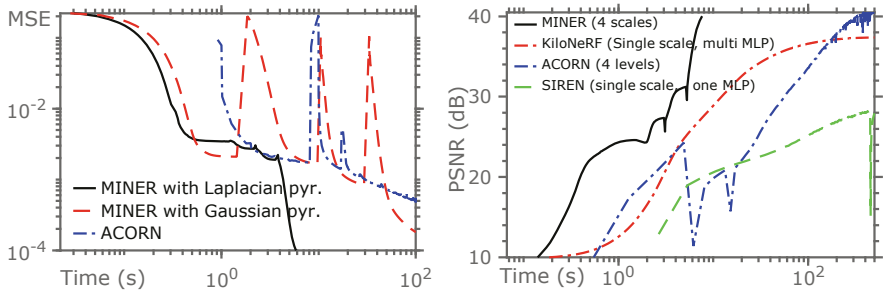
3.2 Training MINER

MINER requires estimation of parameters at each scale and each block. We now present an efficient *sequential* training procedure that starts at the coarsest scale and trains up to the finest scale.

Training at Coarsest Sscale. The training process starts by fitting $\mathrm{I}_{J-1}(\mathbf{x})$, the image at the coarsest scale. We estimate the parameters of each of the MLPs $\mathcal{N}_{J-1}^{(m,n)}$ by solving the objective function,

$$\min_{\mathcal{N}_{J-1}^{(m,n)}} \left\| I_{J-1}\left(\widetilde{\mathbf{x}} + \left(\frac{mH}{2^{J-1}M}, \frac{nW}{2^{J-1}N}\right)\right) - \mathcal{N}_{J-1}^{(m,n)}(\widetilde{\mathbf{x}}) \right\|^2. \tag{6}$$

Let $\widehat{I}_{J-1}(\mathbf{x}_{J-1})$ be the estimate of the image at this stage.



(a) Laplacian vs. Gaussian pyramid **(b)** MINER vs. state-of-art

Fig. 3. Laplacian pyramid enables faster convergence. The plot in (a) shows training error across time for a 2048×2048 image of Pluto. MINER when combined with a Laplacian pyramid offers a significantly faster convergence as the MLPs at finer scale capture orthogonal information. This also results in small jumps in training accuracy that is strongly present when MINER is trained with a Gaussian pyramid, or ACORN. (b) shows PSNR as a function of time for various approaches for a one megapixel Pluto image. MINER achieves higher accuracy at all times, and converges significantly faster than competing approaches. Moreover, the drop in accuracy when changing from coarse to fine scale is less severe for MINER compared to when ACORN re-estimates coordinate decomposition.

Pruning at Convergence. As the training proceeds, some MLPs, particularly for blocks with limited variations, will converge to a target mean squared error (MSE) earlier than the more complex blocks. We remove those MLPs that have converged from the optimization process and continue with the remaining blocks.

Training at Finer Scales. As with the coarsest scale, we continue to fit small MLPs to blocks at each finer scale. For scale $J-2$, the target signal is given by

$$R_{J-2}(\mathbf{x}) = \mathrm{I}_{J-2}(\mathbf{x}) - \mathcal{U}_2(\widehat{\mathrm{I}}_{J-1})(\mathbf{x}/2). \tag{7}$$

We leverage the fact that blocks within each scale occupy disjoint regions and can optimize each MLP independently of one another:

$$\min_{\mathcal{R}_{J-2}^{(m,n)}} \left\| R_{J-2}\left(\widetilde{\mathbf{x}} + \left(\frac{mH}{2^{J-2}M}, \frac{nW}{2^{J-2}N}\right)\right) - \mathcal{R}_{J-2}^{(m,n)}(\widetilde{\mathbf{x}}) \right\|^2. \tag{8}$$

Pruning Before Optimization. Due to the sparseness of gradients of visual signals, we expect a large number of spatial regions to have little to no signal. Nominally, the number of blocks and MLPs double along each dimension at finer scales. However, some blocks may already be adequately represented by the corresponding MLP at the coarser scale. In such a case, we do not assign an MLP to that block and set the estimate of the residue to all zeros. Depending on the frequency content in the image, this decision dramatically reduces the number of total MLP parameters, and thereby the overall training and inference

times. In cases where *a priori* information about residual energy is not available (such as view synthesis from images), we can rely on each block's variance. Blocks with low variance likely converge at the coarser scale and hence can be pruned from training.

4 Experimental Results

Baselines. For fitting to images and 3D volumes, we compared MINER against SIREN [22], KiloNeRF [20], and ACORN [13]. We also compared MINER against convolutional occuppancy networks [19] for 3D volumes. We used code from the respective authors and optimized the training parameters for a fair comparison.

Training Details. We implemented MINER with the PyTorch [18] framework. Multiple MLPs were trained efficiently using the block matrix multiplication function (`torch.bmm`) and hence, we required no complex coding outside of stock PyTorch implementations. All our models were trained on a system unit equipped with Intel Xeon 8260 running at 2.4 Ghz, 128 GB RAM, and NVIDIA GeForce RTX 2080 Ti with 12 GB memory. For all experiments, we excluded any time taken by logging activities such as saving models, images, meshes, and computing intermittent metrics such as PSNR and IoU.

Fitting Images. We split up RGB images into $32 \times 32 \times 3$ patches at all spatial scales. For each patch and at each scale, we trained a single MLP with two hidden layers and sinusoidal activation function [22]. We fixed the number of features to be 20 for each layer. We did not add any further positional encoding. We used the ADAM [8] optimizer with a learning rate of 5×10^{-4} and an exponential decay with $\gamma = 0.999$. At each scale, we trained either for 500 epochs, or until the change in loss function was greater than 2×10^{-7}. We used an ℓ_2 loss function at all scales with no additional prior. We pruned a block from the training pipeline if the block MSE was smaller than 10^{-4}. Similarly, a block was not added at the starting of the training process if the block MSE was smaller than 10^{-4}. The effect of block-stopping threshold is analyzed in the supplementary material.

Figure 3a shows training error for a 4 megapixel (MP) image of Pluto across epochs for MINER by representing a Laplacian pyramid, a Gaussian pyramid, and ACORN. MINER converges rapidly to an error of 10^{-4} compared to other approaches. Moreover, the periodic and abrupt increase in error are more prevalent in Gaussian representation, and ACORN, which further hamper their performance, but not with Laplacian pyramid due to near-orthogonality of signal across scales. Figure 3b shows training error for a 1 MP image for various approaches with a fixed number of parameters (900k). MINER with four scales is nearly two orders of magnitude faster than all approaches. Figure 4 shows the fitting result for a 64 megapixel Pluto image across training iterations. The times correspond to the instances when MINER converged at a given scale. MINER maintains high quality reconstruction at all instances due to the multiscale training scheme and rapidly converges to a PSNR of 40 dB within 50 s. In contrast, ACORN achieves

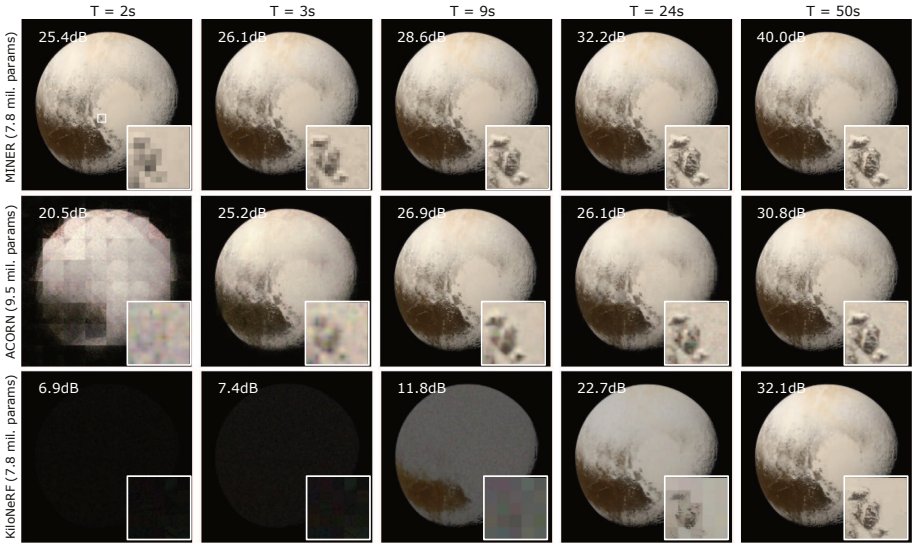


Fig. 4. Image fit over time. The figure compares fitting of the 16 megapixel pluto image at various times during the training process. A distinct advantage of MINER is that the signal is similar to the final output (albeit downsampled) from the starting itself which enables an easy visual debug of the fitting process.

qualitatively good results after 10 s and achieves a PSNR of 30.8 dB after 50 s, and KiloNeRF achieves a qualitatively good result only after 50 s. SIREN Results are not shown in the plot as the first epoch was completed after 4 min. Results with analysis on effect of parameters such as number of scales and patch size is included in supplementary.

Figure 3b shows a plot of PSNR as a function of time for various approaches. We also note that ACORN curve shows significant drop in accuracy as a result of re-computation of coordinate blocks. In contrast, the drop in accuracy for the MINER curve due to scale change is significantly smaller than ACORN. Figure 5 shows results on training a 2 megapixel image with active blocks at all scales. The blocks are concentrated around the high frequency areas (such as the antennae of the grasshopper) as the scale increases from coarse to fine. MINER took less than 10 s to converge to 40 dB fitting accuracy. In contrast, KiloNeRF took 6 minutes to converge and ACORN took 7 min to converge to 40 dB with approximately equal number of parameters.

Figure 6 compares MINER, KiloNeRF, and ACORN in terms of time taken to achieve 36 dB and GPU memory for fitting a 16 MP image of Pluto. For Fig. 6 (a), we used author's implementation of ACORN with default batch size and number of layers and only varied the number of hidden features. For Fig. 6 (b), we set batch size such that up to all pixels were trained simultaneously. Similarly, we set the batchsize for MINER to train all pixels simultaneously to keep comparisons fair. MINER consistently achieves 36 dB faster than competing

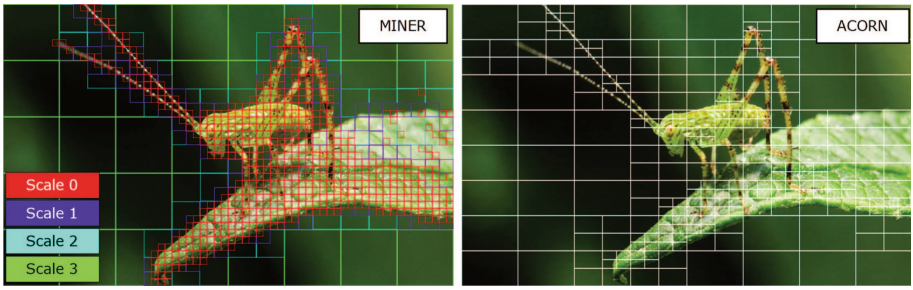


Fig. 5. MINER adaptively selects window sizes. MINER adaptively selects the appropriate scale for each local area resulting in patch sizes that are chosen according to texture variations within the window. The figure above shows a macro photograph of a grasshopper fit by MINER (left image). Large parts of the image such as background have very smooth texture implying that they can be fit accurately at a coarser scale–which translates to large spatial size for low frequency areas. In contrast, area around the antennae are made of high spatial frequencies, which required fitting at finer scales. ACORN provides a similar decomposition (right image) but represents image at only a single image, thereby not being amenable to multiscale analysis.

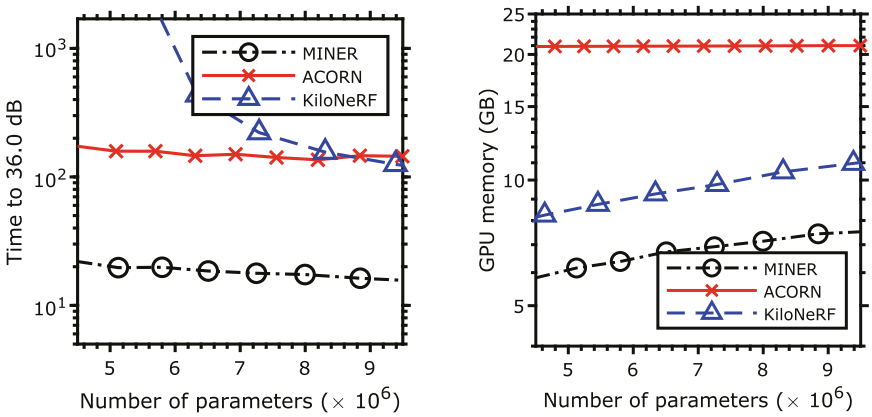


Fig. 6. MINER requires shorter training time and memory footprint. The plot shows the time taken to achieve 36 dB and the GPU memory utilization to fit a 16MP image (Pluto) with ACORN and MINER for varying number of parameters. MINER is an order of magnitude faster than ACORN and requires less than one third of the GPU memory as ACORN–implying MINER is well-suited to train very large models.

methods and requires significantly smaller memory footprint while being able to train on the whole signal at a time, making it highly scalable for large-sized problems.

MINER scales up graciously for extremely large signals. We trained ACORN on a gigapixel image shown in Fig. 7 over 7 scales. We set the number of features

Fig. 7. Fitting gigapixel images. The figure shows the results on fitting a gigapixel image $(20,480 \times 56,420)$ with MINER and ACORN. MINER required 188 million parameters and converged to 38.5 dB in 3.1 h. In contrast, even after 44 h of training, ACORN, which required 175 million parameters, achieved only 32.6 dB.

per each block to be 9, and used a patch size of 32×32. MINER converged to a PSNR of 38.5 dB in 3.1 h and required a total of 164 million parameters. In contrast, after 44 h of training, ACORN converged to only 32.6 dB while using a total of 175 million parameters. We trained both ACORN and MINER on an 11 GB NVIDIA RTX 2080 Ti GPU, which required us to decrease the maximum number of patches for ACORN to 3072 from the original authors' implementation they ran on a 48 GB GPU. MINER also enables compression of

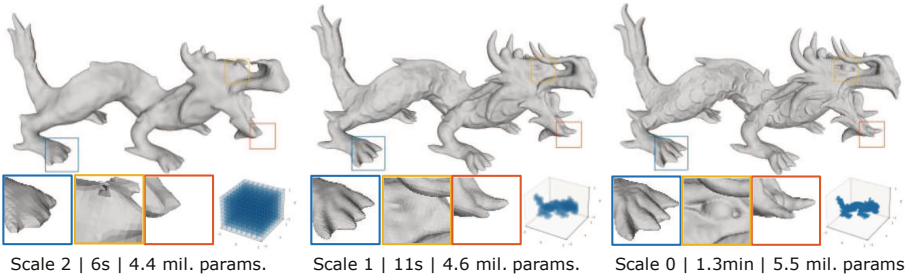


Fig. 8. Active blocks reduce with increasing scale. The figure shows MINER results at the end of training at each scale and the active blocks at each scale. As the iterations progress, only the blocks on the surface of the object remain, which leads to a dramatic reduction in non-zero blocks, and hence the total number of parameters.

the image. Storing the image image as 16-bit `tiff` format required 2.4 GB of disk space. In contrast, MINER required 650 MB with 32 bit precision, implying MINER enables very high compression for images with high dynamic range.

Fitting 3D Point Clouds. cnspired by Convolutional occupancy networks [19], we utilized signed density function where the value was 1 inside the mesh and 0 outside. We sampled a total of one billion points, resulting in a $1024 \times 1024 \times 1024$ occupancy volume. We then optimized MINER over four scales for a maximum of 2000 iterations at each scale. We experimented with logistic loss and MSE and found the MSE resulting in signficantly faster convergence. We divided the volume into disjoint blocks of size $16 \times 16 \times 16$. The learning rate was set to 10^{-3}. We set the number of features to 16 and the number of hidden layers to 2 for MLP for each block at all scales. As with images, we set the per-block MSE stopping threshold to be 10^{-4}, and did not include positional encoding for the inputs. We then constructed meshes from the resultant occupancy volumes using marching cubes [10]. We compared our results against ACORN and convolutional occupancy networks for accuracy and timing comparisons. For ACORN, we used the implementation and the hyperparameters provided by the original authors. For convolutional occupancy networks, we used 200,000 randomly sampled points from the volume as input. Comparisons against screened Poisson surface reconstruction (SPSR) [7], which does not utilize neural networks but requires local normals, is included in the supplementary.

Figure 1 shows reconstructed meshes for the Lucy 3D model at each scale. MINER converges in 22 min to an Intersection over Union (IoU) of 0.999. In the same time, ACORN achieved an IoU of 0.97 with worse results than MINER. ACORN took greater than 7 h to converge to an IoU of 0.999, clearly demonstrating the advantages of MINER for 3D volumes. We also note that MINER required less than a third of the number of parameters as ACORN – this is a direct consequence of using a block-based representation – most blocks outside the mesh and inside the mesh converge rapidly within the first few scales, requiring far fewer representations than a single scale representation. An example of

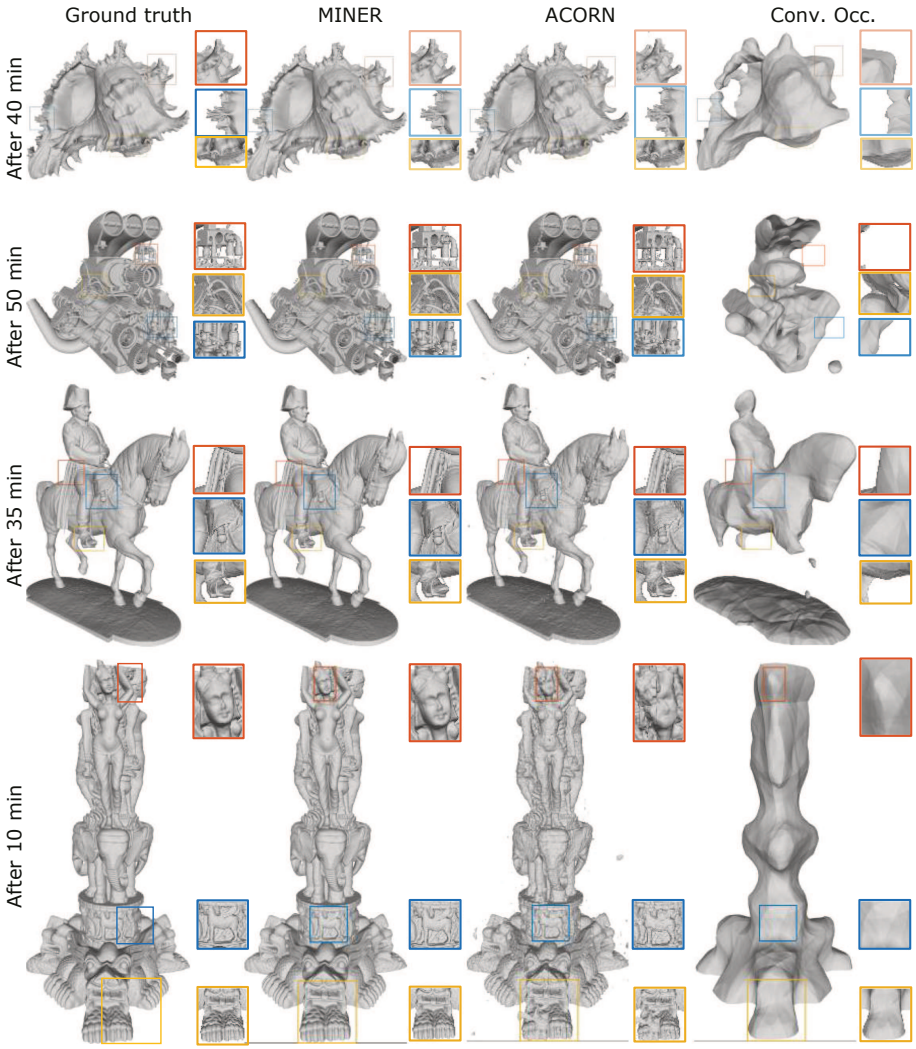


Fig. 9. Comparisons against state-of-the-art for 3D volume fitting. We 3D occupancy fitting for a **fixed duration** with MINER, ACORN, and Convolutional occupancy [19]. The number of parameters of MINER was chosen adaptively according to model complexity. MINER achieves high accuracy in a short duration for arbitrarily complex shapes, which is not possible with prior works, even though some models such as the engine (second row) require significantly more parameters.

active blocks at each scale is shown in Fig. 8. As iterations progress, the number of active blocks after pruning reduce, which in turn results in more compact representation, and fewer parameters. Figure 9 visualizes the meshes fit with various reconstruction approaches for a fixed duration. The time for each experiment

Table 1. Comparison on the Thai Statue 3D Point Cloud. For all experiments, we fixed the batch size to the equivalent of 1024 $16 \times 16 \times 16$ blocks. MINER requires lower training and testing times, similar GPU memory, fewer parameters, and smaller size on disk compared to state-of-the-art techniques. MINER also occupies smaller disk space compared to storing the mesh as a ply file, thereby enabling compression.

	IOU	GPU Mem.	#Params.	Test time	Storage
MINER - scale 3	0.95 (17 s)	1.8 GB	900k	0.02 s	3.5 MB
scale 2	0.97 (42 s)	2.5 GB	1.3 million	0.04 s	4.8 MB
scale 1	0.98 (1.9 min)	5.0 GB	2.8 million	0.16 s	10.6 MB
scale 0	0.99 (6 min)	6.6 GB	9.9 million	0.8 s	37.9 MB
ACORN [13]	0.99 (53 min)	8.0 GB	17 million	18.1 s	68 MB
Conv. Occ [19]	0.82	5.8 GB	160k	–	64 KB
`ply` file	–	–	–	–	180 MB

was chosen to be when MINER achieved an IoU of 0.999. MINER has superior reconstruction quality compared to ACORN and convolutional occupancy networks [19]. Table 1 compares IoU after a fixed time, GPU memory for training, number of parameters, testing (inference) time, and disk space for MINER at various scales, and competing approaches. The memory usage of ACORN increased from 3.9 GB at the start (with no further splitting) to 8 GB, which we reported. MINER achieves high accuracy (IoU) within 17 s at the coarsest scale where GPU utilization, number of parameters (and hence size on disk) are low. At the finest scale, MINER achieves very high accuracy, and requires fewer parameters, thereby enabling training on very large meshes. Moreover, MINER occupies a third of the size on disk compared to a standard `ply` file, thereby enabling mesh compression.

5 Conclusions

We have proposed a novel multi-scale neural representation that trains faster, requires same or fewer parameters, and has lower memory footprint than state-of-the-art approaches. We demonstrated that the advantages of a Laplacian pyramid including multiscale and sparse representation enable computational efficiency. We showed that leveraging self-similarity across scales is beneficial in reducing training time drastically while not affecting the training accuracy. MINER naturally lends itself to rendering where level-of-detail is of importance including representation and mipmapping for texture mapping. MINER can be combined with fast, multiscale rendering approaches [31] to achieve real time neural graphics. With the low computational complexity and fast training and inference time, MINER opens avenues for rendering extremely large and complex geometric shapes that was previously impractical.

Acknowledgements. This work was supported by NSF grants CCF-1911094, EEC-1648451, IIS-1838177, IIS-1652633, and IIS-1730574; ONR grants N00014-18-12571, N00014-20-1-2534, and MURI N00014-20-1-2787; AFOSR grant FA9550-22-1-0060; and a Vannevar Bush Faculty Fellowship, ONR grant N00014-18-1-2047.

References

1. Chabra, R., et al.: Deep local shapes: learning local SDF priors for detailed 3D reconstruction. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12374, pp. 608–625. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58526-6_36
2. Chen, H., He, B., Wang, H., Ren, Y., Lim, S.N., Shrivastava, A.: Nerv: Neural Representations for Videos. Adv. Neural Info, Processing Systems (2021)
3. Chen, Y., Liu, S., Wang, X.: Learning continuous image representation with local implicit image function. In: IEEE Computer Vision and Pattern Recognition (CVPR) (2021)
4. Chien, C.H., Aggarwal, J.K.: Volume/surface octrees for the representation of three-dimensional objects. Comput. Vision Graph. Image Process. **36**(1), 100–113 (1986)
5. Do, M.N., Vetterli, M.: Framing pyramids. IEEE Trans. Signal Process. **51**(9), 2329–2342 (2003)
6. Hertz, A., Perel, O., Giryes, R., Sorkine-Hornung, O., Cohen-Or, D.: Sape: spatially-adaptive progressive encoding for neural optimization. Adv. Neural Info. Process. Syst. **34**, 8820–8832 (2021)
7. Kazhdan, M., Hoppe, H.: Screened poisson surface reconstruction. ACM Trans. Graph. **32**(3), 1–13 (2013)
8. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: International Conference Learning Representations (2015)
9. Kuznetsov, A., Mullia, K., Xu, Z., Hašan, M., Ramamoorthi, R.: NeuMIP: Multi-resolution neural materials. ACM Trans. Graphics **40**(4), 1–13 (2021)
10. Lorensen, W.E., Cline, H.E.: Marching cubes: a high resolution 3D surface construction algorithm. ACM SIGGRAPH **21**(4), 163–169 (1987)
11. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Intl. J. Comput. Vision **60**(2), 91–110 (2004)
12. Mairal, J., Sapiro, G., Elad, M.: Multiscale sparse image representation with learned dictionaries. In: IEEE International Conference Image Processing (ICIP), vol. 3, pp. III-105 (2007)
13. Martel, J.N., Lindell, D.B., Lin, C.Z., Chan, E.R., Monteiro, M., Wetzstein, G.: Acorn: Adaptive coordinate networks for neural scene representation. arXiv preprint arXiv:2105.02788 (2021)
14. Meagher, D.: Geometric modeling using octree encoding. Comput. Graph. Image Process. **19**(2), 129–147 (1982)
15. Mehta, I., Gharbi, M., Barnes, C., Shechtman, E., Ramamoorthi, R., Chandraker, M.: Modulated periodic activations for generalizable local functional representations. In: IEEE International Conference Computer Vision (ICCV) (2021)
16. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: IEEE European Conf. Computer Vision (ECCV) (2020)
17. Park, J.Y., Wakin, M.B.: A multiscale framework for compressive sensing of video. In: Picture Coding Symposium (2009)

18. Paszke, A., et al.: Pytorch: An imperative style, high-performance deep learning library. In: Advance Neural Information Processing Systems (2019)
19. Peng, S., Niemeyer, M., Mescheder, L., Pollefeys, M., Geiger, A.: Convolutional occupancy networks. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12348, pp. 523–540. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58580-8_31
20. Reiser, C., Peng, S., Liao, Y., Geiger, A.: Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. arXiv preprint arXiv:2103.13744 (2021)
21. Shapiro, J.M.: Embedded image coding using zerotrees of wavelet coefficients. In: Fundamental Papers in Wavelet Theory, pp. 861–878 (2009)
22. Sitzmann, V., Martel, J., Bergman, A., Lindell, D., Wetzstein, G.: Implicit neural representations with periodic activation functions. Adv. Neural Info, Processing Systems (2020)
23. Srinivasan, P.P., Deng, B., Zhang, X., Tancik, M., Mildenhall, B., Barron, J.T.: Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In: IEEE Computer Vision and Pattern Recognition (CVPR) (2021)
24. Sulam, J., Ophir, B., Elad, M.: Image denoising through multi-scale learnt dictionaries. In: IEEE International Conference Image Processing (ICIP) (2014)
25. Sun, Y., Liu, J., Xie, M., Wohlberg, B., Kamilov, U.S.: Coil: Coordinate-based internal learning for imaging inverse problems. arXiv preprint arXiv:2102.05181 (2021)
26. Takikawa, T., et al.: Neural geometric level of detail: Real-time rendering with implicit 3d shapes. In: IEEE Computer Vision and Pattern Recognition (CVPR) (2021)
27. Tancik, M.: Fourier features let networks learn high frequency functions in low dimensional domains. Adv. Neural Info, Processing Systems (2020)
28. Thévenaz, P., Unser, M.: Optimization of mutual information for multiresolution image registration. IEEE Trans. Image Processing **9**(12), 2083–2099 (2000)
29. Weber, J., Malik, J.: Robust computation of optical flow in a multi-scale differential framework. Intl. J. Comput. Vision **14**(1), 67–81 (1995)
30. Yifan, W., Rahmann, L., Sorkine-Hornung, O.: Geometry-consistent neural shape representation with implicit displacement fields. arXiv preprint arXiv:2106.05187 (2021)
31. Yu, A., Li, R., Tancik, M., Li, H., Ng, R., Kanazawa, A.: Plenoctrees for real-time rendering of neural radiance fields. arXiv preprint arXiv:2103.14024 (2021)