
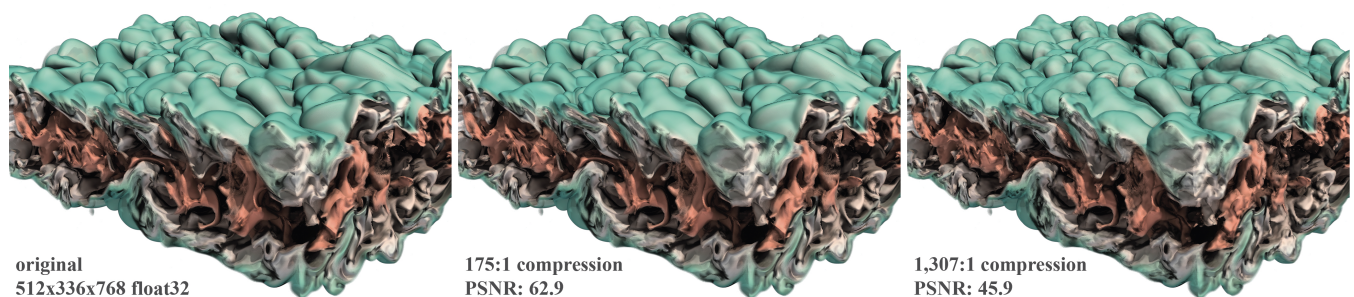


# Compressive Neural Representations of Volumetric Scalar Fields

Y. Lu<sup>1</sup>, K. Jiang<sup>2</sup>, J. A. Levine<sup>2</sup> , and M. Berger<sup>1</sup>

<sup>1</sup>Department of Electrical Engineering and Computer Science, Vanderbilt University, USA

<sup>2</sup>Department of Computer Science, University of Arizona, USA



**Figure 1:** Our approach represents a scalar field as a neural network that conditions on a point in the field's domain and produces a scalar value. We obtain highly compressive volume representations in this manner; here showing two levels of compression (middle, right) for the jet volume (left), where even for extreme compression ratios (right, 1307 : 1), predominant features of the volume are preserved.

## Abstract

We present an approach for compressing volumetric scalar fields using implicit neural representations. Our approach represents a scalar field as a learned function, wherein a neural network maps a point in the domain to an output scalar value. By setting the number of weights of the neural network to be smaller than the input size, we achieve compressed representations of scalar fields, thus framing compression as a type of function approximation. Combined with carefully quantizing network weights, we show that this approach yields highly compact representations that outperform state-of-the-art volume compression approaches. The conceptual simplicity of our approach enables a number of benefits, such as support for time-varying scalar fields, optimizing to preserve spatial gradients, and random-access field evaluation. We study the impact of network design choices on compression performance, highlighting how simple network architectures are effective for a broad range of volumes.

## CCS Concepts

• **Human-centered computing** → Visualization; • **Computing methodologies** → Neural networks; Image compression;

## 1. Introduction

The visualization of large-scale field-based data is a fundamental component to many post hoc analyses. Field data, in particular scalar fields, often arise from numerical simulations of scientific phenomena, which require high (3D) spatial and temporal resolution to accurately resolve domain-specific features of interest. The size of such large-scale data presents a number of challenges for visualization, ranging from bandwidth constraints, disk storage, data accessibility, and interactivity. Compression of volumetric scalar fields remains an important tool that can help mitigate these challenges. For visualization purposes, the data is often so large that *lossy* compression methods are required to enable analysis. The main purpose of lossy methods is to obtain compact volumetric

representations in which features of the data remain visually perceptible, at the cost of sacrificing some data precision.

Within the visualization community, the lossy compression of volumetric scalar fields has largely been based on transforms that admit compressive representations, where transform coefficients can be discarded and/or aggressively quantized with minimal loss in data fidelity. A commonality to these methods is the reliance on a rectilinear grid for which predefined bases may be constructed, e.g. Fourier [YL95] or Wavelet bases [WMB\*11], or where data-dependent bases may be derived [SMP13, BRP16]. However, the success of these methods heavily depends on whether a provided scalar field is a good match for the assumptions made by the transform, e.g. the field is largely composed of low-

frequencies for Fourier bases, or admits a low-rank decomposition for factorization-based transforms [BRLP19]. In practice, these assumptions are satisfied to varying degrees, leading to variability in compression ratios and approximation quality amongst existing methods.

In this work we propose a learning-based method for compressing scalar fields. We leverage neural networks that are designed to map a continuously-defined position from the domain to a scalar value in the range [PFS\*19, MON\*19]. The use of neural networks places little assumptions on data characteristics; instead, the network learns what is necessary to approximate the given field. In particular, by limiting the capacity of the network, such that the number of weights of the network is less than the volume resolution, we obtain compressive volume representations that are optimized to best approximate the field at its sampled values. Hence, the neural network *is* the compressed volume representation – the level of compression, in part, follows from the number of network weights, and the original sampled field can be reconstructed by evaluating the neural network at the given positions.

Our neural network design is conceptually straightforward. We build on the approach of SIREN [SMB\*20], wherein fully-connected layers and periodic activation functions (using sinuoids) comprise the network architecture. Periodic activation functions, coupled with careful initialization for stable optimization, have been recently shown to outperform more advanced architectures, e.g. frequency-based embeddings with ReLU activations [MST\*20], for a variety of representation tasks. We illustrate how residual connections that preserve SIREN activation distributions lead to networks that are robust across architecture choices. Further, we observe that the learned, per-layer, weights of SIREN are distributed in such a manner that they may be quantized to a small number of bits, with minimal impact on performance.

By limiting network capacity, and performing weight quantization, we obtain highly compact representations of scalar fields, as demonstrated in Fig. 1. Further, the conceptual simplicity of our approach provides a number of benefits. Due to the use of periodic activation functions, we can target the optimization of higher-order derivatives, as studied in Sitzmann et al. [SMB\*20]. In particular, when spatial gradients are available, we can use this information to regularize the network. This leads to better-behaved scalar fields whose isosurfaces are less noisy under high compression ratios. We also show how it is straightforward to adapt our approach to compressing time-varying scalar fields simply by adjusting the network inputs. Furthermore, the networks enable field access at arbitrary points in the domain, eschewing the reconstruction of the entire volume at once – we show how this can benefit visualization applications such as volume rendering.

Summarizing, the main contributions of our approach are:

1. A compression technique for volumetric scalar fields based on implicit neural representations.
2. An evaluation of this compression approach, directly comparing it to a recent state-of-the-art technique (tthresh [BRLP19]) as well as exploring implementation design choices; and
3. Experimental results on compressing volumes, including studying gradient preservation and preserving time-varying data.

## 2. Related Work

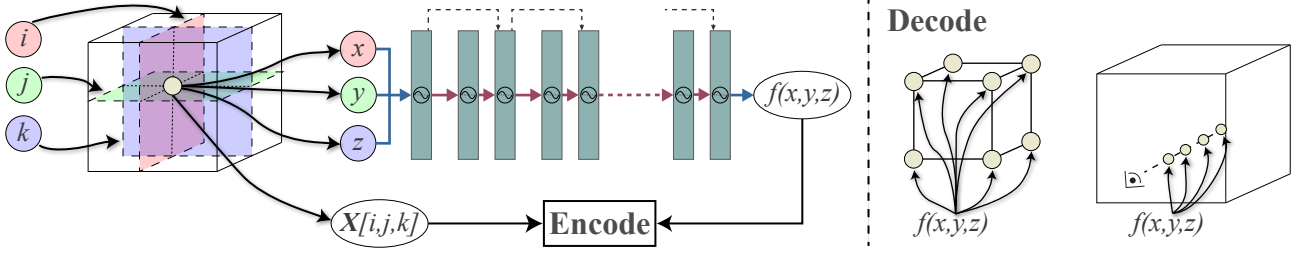
### 2.1. Compression of Volumetric Data

Lossy compression of large scale volumes has been an important topic in recent years, particularly as our ability to simulate and acquire data grows. Many of the early works in this area focused on using discrete cosine transforms [YL95] or wavelet-based compression [GWGS02, IP99, Mur93, WMB\*11], often specifically focusing on interactive rendering applications [RGG\*13, SW03]. By using such transforms, these techniques enable compression by allowing the end user to separate features within a transformed space (i.e. removing high frequencies while preserving low frequencies) and ultimately sparsify the data representation. ZFP performs transformations customized at the block level to achieve compression with fast I/O access [Lin14]. More recent techniques, such as TAM-RESH [SMP13] and TTHRESH [BRLP19] employ tensor decomposition [BRP16]. TTHRESH is particularly notable as a recent state-of-the-art compression technique that we directly compare our results against in this work.

Other techniques focus on data fitting as the workhorse for compression. For example, ISABELA [LSE\*11] and SZ [DC16, TDCC17] use curve fitting to approximate the input data. Liang et al. outperform SZ and ZFP using an improved Lorenzo prediction [LDT\*18]. Data fitting is related to our neural network based approach, although the method to achieve the fit is different. Using data models can be a powerful mechanism for providing a compression technique. Peterka et al. [PNG\*18] study the use of nonuniform rational B-spline functions (NURBS) for compressive volume representations via adaptive refinement of control points. This work shows the benefit of smooth representations for analytically-defined gradients, however an important distinction in our approach is that we can directly optimize for gradients. Other examples focus on models that preserve specific aspects of the data, e.g. topological features [SPCT18], graph-based models like SQ [IKK12], and dictionaries [DMG20]. COVRA uses an octree decomposition to model data blocks which are compressed at multiple resolutions [GIGM12]. Data fitting broadly considers the tradeoff between how many samples are used and how much precision is devoted to each sample. Recently, Hoang et al. provide a precision-resolution tree that is used for efficient selection of either fewer data samples or data samples encoded with fewer bits [HSB\*20].

### 2.2. Deep Learning for Volume Representation

Recently, numerous researchers have deployed deep learning and used neural networks to enable a variety of visualization tasks. Berger et al. utilize generative adversarial networks (GANs) to analyzing the space of visual parameters in a volume renderer, providing a new interface for transfer function design [BLL18]. He et al. generalize the idea of learning conditioned on parameter spaces to include ensembles of volumes with InSituNet [HWG\*19]. Han et al. encode multivariate volume data for variable-to-variable translation [HZX\*20]. Jakob et al. consider the problem of interpolation using neural networks defined on a large ensemble of vector fields [JGG20]. While the applications are different, all of these works share a common thread in that they construct representations that blend visualization tasks (e.g. rendering, interpolation) with data representations.



**Figure 2:** An overview of our compression method, split into the steps of encoding (left) and decoding (right). We encode a provided scalar field with a coordinate-based neural network, where the network  $f$  is tasked with predicting the field's value  $\mathbf{X}[i, j, k]$  given coordinates  $(x, y, z)$ . In the network diagram, red arrows represent fully-connected layers whose weights are quantized, blue arrows are unquantized fully-connected layers, and dashed black arrows represent residual connections. Decoding amounts to function evaluation, where we may reconstruct the field at grid vertices, or arbitrary positions, and thus perform volume rendering with the network in lieu of the sampled field.

Closely related to compression is the task of super-resolution, for which many techniques have been developed in the computer vision communities. More recently, researchers have shown that deep learning can be used to achieve super-resolution on volumetric data. Xie et al. developed tempoGAN for super-resolution of volumetric fluids in computer graphics [XFCT18]. Weiss et al. developed new techniques for isosurface rendering that similarly used deep learning for super-resolution [WCTW19]. Han and Wang developed neural networks for super-resolution in temporal [HW19] and spatial [HW20] super-resolution, and Guo et al. achieved spatial super-resolution in vector fields [GYH\*20]. Notably, many of these techniques are tested against possible compression techniques, but none of the above were specifically designed for compression. Thus, while they are compressive, their main strengths lie elsewhere.

Recently, the machine learning community has explored using deep learning to construct implicit representations, which can be used to model a variety of data. We utilize a similar approach in our network design, and as seen in recent works this can achieve superior performance. Park et al. developed DeepSDF for representing shape using signed distance fields [PFS\*19]. Recent extensions to this work include overfit neural networks [DNJ20] and DualSDF [HAESB20] that allow for tradeoffs in the representation precision and flexibility. Similarly, Chen and Zhang use implicit fields for generative shape modeling [CZ19]. Alternatively, instead of training to produce a signed distance field, Mescheder et al. propose to construct occupancy fields [MON\*19]. Besides just representing shape, other recent applications include texture synthesis [OMN\*19] and view synthesis [MST\*20]. As this is an emerging area, we are still discovering new ways to improve these networks. Notable recent works that informed our network design include using periodic activations [SMB\*20] and Fourier feature mappings [TSM\*20]. Our approach is distinct from prior works in that we investigate how to make implicit neural representations compressive for arbitrary scalar fields [DNJ20, SMB\*20, TSM\*20].

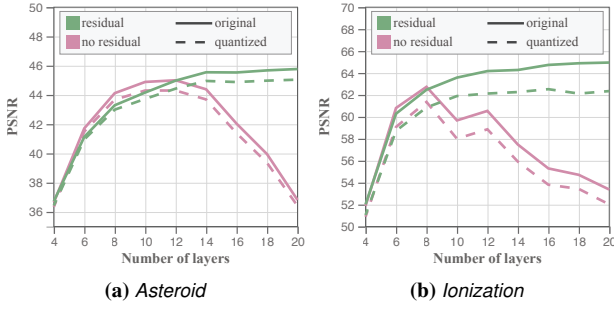
### 3. Compression Method

The basic idea behind our compression approach is to represent a provided scalar field as a learned function, one that is parameterized by a set of weights whose size is smaller than the field's

size. Fig. 2 shows an overview. Specifically, we assume the input to our method is a scalar field sampled on a regular grid, which we treat as a  $d$ -tensor  $\mathbf{X} \in \mathbb{R}^{s_1 \times s_2 \times \dots \times s_d}$ . When  $d = 3$  this represents a scalar field whose domain is a 3D Cartesian coordinate system, while  $d = 4$  may represent a time-varying scalar field in 4 dimensions. We associate each sample of the field with an indexing tuple of integers  $\mathbf{i} = (i_1, i_2, \dots, i_d)$ , an integer for each dimension, such that  $\mathbf{X}[i_1, i_2, \dots, i_d]$  returns the value of the field at this sample, and further, assume this index is associated with a real-valued  $d$ -dimensional point from the field's domain,  $\mathbf{p}_i \in \mathbb{R}^d$ .

Our goal is to learn a function  $f_\Theta : \mathbb{R}^d \rightarrow \mathbb{R}$ , such that for a given sample  $\mathbf{i} = (i_1, i_2, \dots, i_d)$  and corresponding point  $\mathbf{p}_i$ , we would like  $f_\Theta(\mathbf{p}_i)$  to be as close to  $\mathbf{X}[i_1, i_2, \dots, i_d]$  as possible. The vector  $\Theta \in \mathbb{R}^m$  denotes the function's set of parameters. Assuming its size  $m$  is less than the size of the field, defined as  $C = \prod_{j=1}^d s_j$ , then we can obtain a representation of the field that (a) serves as a good approximation to  $\mathbf{X}$ , and (b) requires smaller storage.

What specific form should the function take? In order to handle complex volumetric fields typically produced in numerical simulations, we require functions that place as few assumptions as possible on the characteristics of the fields, e.g. smoothness, spectral properties, or sampling rate. To this end, we define the functions as deep neural networks, building off of recent work in implicit neural representations for shape modeling [PFS\*19, MON\*19, SMB\*20]. Notably, these methods can model fine-grained details, e.g. intricate geometric structures of shapes, only limited by the capacity of the network rather than the resolution of some underlying regular grid, as is commonly used with convolutional networks [HW19]. We argue that such coordinate-based neural networks are a good fit for arbitrary volumetric fields, not just shape-based data. By framing compression as optimization, we allow the network to learn what is necessary for approximating the sampled field under a compression budget. In this paper we study how to effectively construct such functions, considering the trade-offs in approximation quality and level of compression. At a high-level, our approach follows a standard compression setup, comprised of 2 steps: an *encode* step to find the function, and a *decode* step that allows us to reconstruct the sampled field.



**Figure 3:** We show the impact of residual connections (green) vs. non-residual connections (red) as a function of network depth, as well as the impact of network weight quantization (dashed lines), fixing the number of network weights  $m$  to be  $\frac{C}{50}$ . We find that residual connections are more robust to depth and improve performance, while weight quantization leads to a small drop in performance, but is volume-dependent.

### 3.1. Neural Encoder

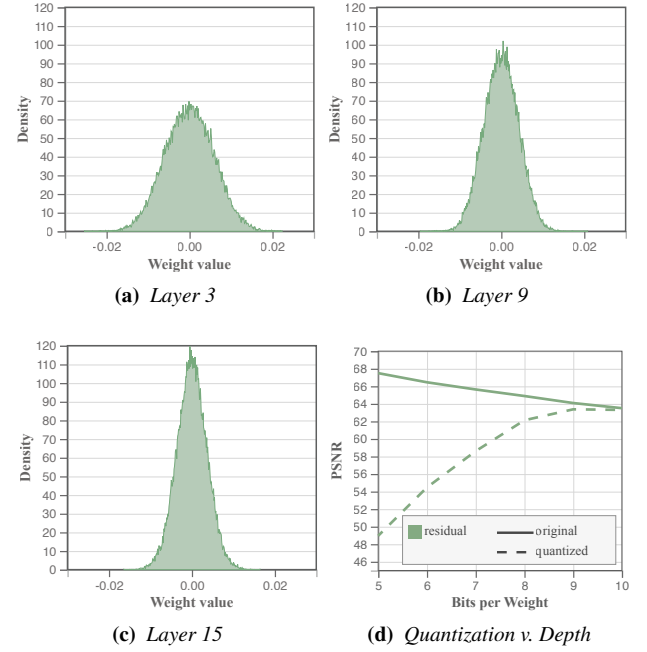
The purpose of the encode step is to train the neural network to accurately model the field at its provided samples. More specifically, we aim to minimize the following mean-squared error loss:

$$\min_{\Theta} \sum_i (f_{\Theta}(\mathbf{p}_i) - \mathbf{X}[i_1, i_2, \dots, i_d])^2. \quad (1)$$

In practice, the loss function is optimized by performing stochastic gradient descent, assembling minibatches at each iteration of training by randomly selecting samples from the provided field.

What remains is the design of the network. We opt for simplicity in network architectures, to avoid the brittleness of hyperparameter tuning and support a diverse set of volumetric fields. To this end, we build off of the approach of SIREN [SMB\*20], where a neural network is defined by a sequence of fully-connected layers that use sinusoidal activation functions – no normalization, e.g. batch normalization [IS15], is necessary for successful training. SIREN has several benefits. First, as demonstrated by Sitzmann et al., training is ensured stable through a principled initialization scheme that controls for the distribution of activations at layers, namely that all layers are arcsine-distributed. Secondly, the use of sinusoidal activations admits functions that are  $C^\infty$  differentiable. Our approach takes advantage of both of these properties.

Given the network design of SIREN, to setup the network we need only define the fully-connected layers. To this end, our method accepts two inputs: the number of layers  $l$ , and the total number of network weights  $m$ . We derive an integer  $k$  such that the first weight matrix has size  $k \times d$ , the last weight matrix has size  $d \times 1$ , and all intermediary weight matrices are of size  $k \times k$ , such that the total number of weights of the network is approximately  $m$ . Bias vectors are similarly derived from inputs  $m$  and  $l$ . All weights are of 32-bit floating-point precision. The resulting set of weight matrices and bias vectors, collectively, represents our vector of parameters  $\Theta$ , e.g. the compressed representation. Note that it is also possible to vary the sizes of the weight matrices, e.g. to gradually increase the number of hidden units as a function of layer depth. Experi-



**Figure 4:** In (a-c) we show the distribution of weights for our network trained on the Ionization volume for layers 3, 9, and 15, number of weights  $m = \frac{C}{50}$ . We find that the distribution is approximately zero-mean normal, with slight variations, hence amenable to quantization. In (d) we vary the number of bits used for quantization, adjusting the number of weights  $m$  to ensure an approximately equal compression ratio. We find that 9 bits is a good trade-off between number of network weights, and level of quantization.

mentally, we found this made little difference, thus for simplicity we chose a fixed number of hidden units in intermediary layers.

The number of layers,  $l$ , should be large enough to handle the modeling of arbitrarily complex fields. In practice, however, setting  $l$  too large can be detrimental, posing challenges for training due to vanishing gradients. As an example, we show results for two different volumes in Fig. 3 (red solid lines), varying the number of layers, where the quality of the volume is measured through peak signal-to-noise ratio (PSNR), the mean-squared error normalized by the data range. Note that, at a certain point, as the model depth increases performance decreases. Further, the network depth that leads to highest PSNR varies between the two volumes.

In order to ensure our models are robust with respect to the number of layers across different volumes, we enrich SIREN with residual connections [HZRS16]. Specifically, we modify the identity mappings of He et al. [HZRS16] to ensure that the distribution of activations in each layer are within the range  $[-1, 1]$ , in accordance with the assumptions of SIREN [SMB\*20]. Each residual block in our network takes the following form, omitting bias vectors for clarity (c.f. Fig. 2 dashed black arrows):

$$\mathbf{a}_{i+1} = \frac{1}{2} \left( \mathbf{a}_i + \sin \left( \mathbf{M}_{i+1}^2 \sin \left( \mathbf{M}_{i+1}^1 \mathbf{a}_i \right) \right) \right), \quad (2)$$



where  $\mathbf{a}_i$  represents a vector of activations from a previous block  $i$ ,  $\mathbf{a}_{i+1}$  is a vector of activations at the next block  $i+1$ , and  $\mathbf{M}_{i+1}^1$  and  $\mathbf{M}_{i+1}^2$  are a pair of learnable matrices associated with block  $i+1$ . In Fig. 3 (green solid lines) we show the effect of residual connections – note we treat each block as containing 2 layers. Note that (a) training is more stable as the network depth increases, and (b) we obtain a boost in performance over non-residual connections. Consequently, unless otherwise stated, all of our networks employ 8 residual blocks.

### 3.1.1. Network Weight Quantization

Thus far, the compression ratio of our method can be expressed as  $\frac{C}{m}$ , assuming the network weights and volume are of the same precision. However, further compression can be gained through quantizing the network weights. Weight quantization has been considered for classification-based convolutional networks [HMD15], but to the best of our knowledge, not yet studied for coordinate-based MLPs. For our SIREN-based network, we make several observations:

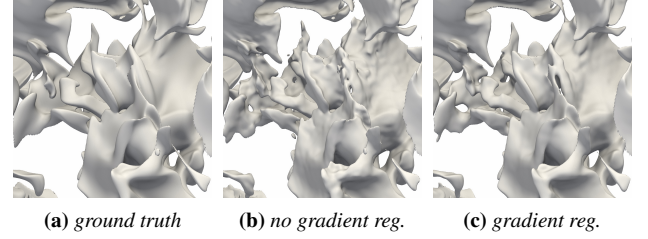
1. The first and last layers of the network are parameterized by small weight matrices, as discussed above. Experimentally, we found that they require high precision for good performance and thus we do not quantize the matrices (c.f. Fig. 2 blue arrows).
2. Instead, most of the parameters of our network exist in intermediary layers, parameterized by matrices of  $k \times k$ . It is these matrices that we quantize (c.f. Fig. 2 red arrows).
3. Since the distribution of activations in SIREN take a specific form, we find that the distribution of weights in each layer take on, approximately, a normal distribution, with some small variations. Fig. 4(a)-(c) demonstrates this for three different layers, where empirically, we find that deeper layers have lower spread, and a higher concentration of values around zero.

Motivated by these observations, we employ the technique of Han et al. [HMD15] and cluster the weights, individually, for each intermediary layer using k-means. We set the number of clusters to be the precision at which we would like to represent the weights, namely if we wish to represent the weights with a precision of  $b$  bits, then the number of clusters will be set to  $2^b$ . After clustering, we assign each weight its corresponding  $b$ -bit index, and store these, alongside the corresponding floating point-precision cluster centers, as the quantized weight representation.

What should the number of bits,  $b$ , be set to in practice? In Fig. 3 we show the two network options with their corresponding quantized representations (as dashed lines), setting  $b$  to 8. We find that the drop in performance in quantization is dependent on the specific volume, with the ionization volume most impacted. Thus, we cannot be too aggressive in quantization, yet there exists a size trade-off in the precision at which to represent weights, and the number of weights to assign to the network. Based on this, in Fig. 4(d) we plot the PSNR for a sequence of networks that all have approximately the same size, but consider both the number of weights and weight quantization. We increase the number of bits used for quantization along the x-axis, and compensate by decreasing the number of weights of the network. We observe that at 9 bits, e.g. 512 clusters, the network takes a minimal drop in PSNR – similar results

**Table 1:** We compare the impact of gradient regularization (“grad”) without using such regularization (“no grad”) for the jet volume, for a compression ratio of 677 : 1. We find this regularization yields gradients closer to the target (Grad PSNR) with minimal drop in field approximation (PSNR).

Loss	PSNR	FD-Grad PSNR	Net-Grad PSNR
no grad	51.8	52.0	50.3
grad	51.6	54.7	55.5



**Figure 5:** We show isosurfaces for the reconstructed jet volume, comparing with, and without, gradient regularization (c.f. Table 1) in the loss. Note that gradient regularization helps to suppress erroneous high-frequency details.

are observed in different volumes. Consequently, all networks in the paper represent weights of intermediary layers with 9 bits.

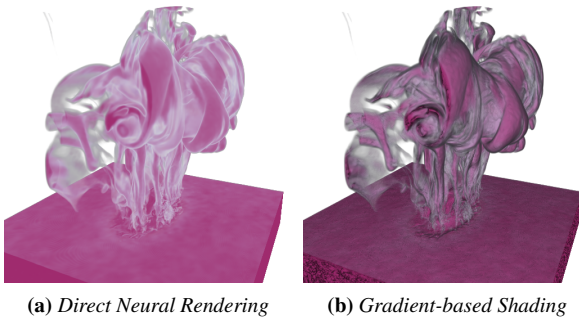
### 3.1.2. Gradient Regularization

As previously mentioned, the use of sinusoidal activation functions in our network implies that we can take arbitrary-order derivatives of the function. Thus we can optimize for the derivatives of the function, in addition to just the provided field values. This allows us to obtain compressed fields whose higher-order information is preserved. Specifically, we modify our loss in Eq. 1 to ensure that the spatial gradient of the function is close to provided gradients of the original scalar field:

$$\min_{\Theta} \sum_i (f_{\Theta}(\mathbf{p}_i) - \mathbf{X}[i_1, i_2, \dots, i_d])^2 + \lambda \|\nabla f_{\Theta}(\mathbf{p}_i) - \mathbf{X}'[i_1, i_2, \dots, i_d]\|_2^2, \quad (3)$$

where  $\mathbf{X}'$  denotes the (provided) scalar field gradient, and we set  $\lambda = 0.05$ , which we found to give a good balance between the gradient and scalar value fit. In some cases,  $\mathbf{X}'$  may be accompanied by the field, e.g. within FEM numerical simulations gradients are often computed from the finite element basis. In other cases, we may numerically approximate gradients using finite differencing schemes, however for certain volumes, e.g. acquired medical images, the estimated gradients may not be reliable for regularization.

To show the benefit of gradient regularization we train 2 networks – one with gradient regularization and one without – to compress the jet volume, under a compression ratio of 677 : 1 for each. Namely, each network is trained with number of weights  $m = \frac{C}{200}$ , along with weight quantization. We use central differencing to numerically estimate gradients from the provided field.



**Figure 6:** Our neural network can be directly used for volume rendering, rather than reconstructing the volume, shown here for *asteroid* for (a) a basic volume renderer where our network is evaluated on-demand during ray marching, and (b) the computation of network gradients to enable direct illumination.

We compare the compression results in terms of the PSNR of the original field, as well as PSNR of the gradient field, following Han et al. [HTZ\*19]. The numerically estimated gradient from the provided scalar field is taken as the target gradient field, while we compare to 2 forms of gradients from the decompressed volumes: central differencing of the reconstructed volume (denoted FD-Grad), as well as the analytical gradients obtained by differentiating the network (denoted Net-Grad).

Table 1 summarizes the results. We find that incorporating gradient regularization leads to a small decrease in PSNR of the scalar field, but leads to a boost in performance in capturing the target gradient. Interestingly, without gradient regularization, analytical gradients from the network (Net-Grad PSNR) suffer quite a bit relative to numerically-estimated gradients (FD-Grad PSNR). From a visualization perspective, inaccurate gradients can have a large impact on isosurfaces of the scalar field, whose normals point in the same direction as the gradients. We highlight this phenomenon in Fig. 5. Note that without gradient regularization, we obtain surfaces with erroneous high-frequency details, whereas gradient regularization leads to smoother surfaces, reflective of ground truth.

### 3.2. Neural Decoder

Once the network has been trained, and the volume encoded, decoding is rather straightforward, as it amounts to evaluating the neural network. To reconstruct the scalar field we simply evaluate our network at all grid vertices within the volume. This evaluation is relatively efficient given the data parallelism of the network, e.g. a sequence of matrix multiplications and sin function evaluations.

Beyond reconstruction, a key advantage of neural scalar field representations is the support for random access function evaluation. We may evaluate our function at *any* position, not just grid vertices, hence the network serves as an interpolant. Furthermore, we can compute higher-order information from the network directly, eliminating the need to numerically estimate derivatives. These features allow us to use the network *directly* for visualization purposes, rather than reconstructing the full volume a priori. As an illustration,

we have developed a neural volume renderer with our network, where at each step of ray marching, we evaluate our network at all current ray positions to obtain function values, see Fig. 6a for an illustration. Moreover, it is trivial to support direct illumination via the computation of function gradients as we ray march, subsequently normalized for shading purposes. We highlight this feature in Fig. 6b. The quality of the volume-rendered images suggests that our network serves as a good interpolant, not merely overfitting to the sampled field's values.

## 4. Experiments

To evaluate our approach we have run our method on a variety of datasets under different levels of compression; Table 2 summarizes them. Specifically, from the Johns Hopkins Turbulence Database (JHTD) [LPW\*08], *mhd\_p* and *mhd\_bx* are respectively pressure and x-coordinate magnetic fields from a magneto-hydrodynamic isotropic turbulence simulation, and *isotropic\_p* is a pressure field from a forced isotropic turbulence simulation. The ionization volume is a temperature field from an ionization front instability simulation [WN], the jet volume is mixture fraction from a simulation of jet flames [GBG\*14], the magnetic volume is from a magnetic reconnection simulation [GLDL14], and *rt* is from a Rayleigh-Taylor instability simulation [CCM04]. Volumes for JHTD and *rt* are formed by taking centered spatial crops to access the original field values where the central portion of the simulations take place.

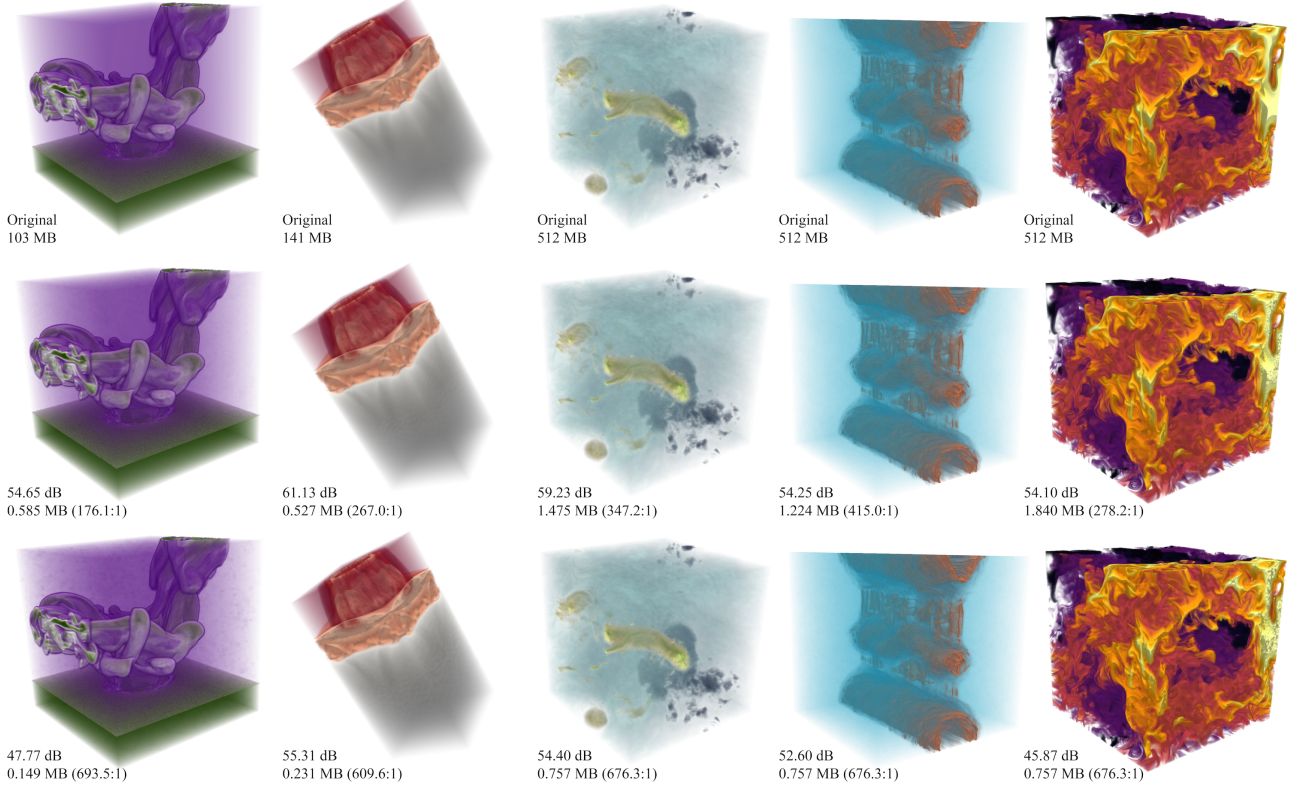
Furthermore, we have evaluated our method on two time-varying volumes *isabel* and *asteroid* (Section 4.2). The dataset *isabel* corresponds to hurricane Isabel's "QVapor" field [WBK04] of its first twelve time steps. Note that we cropped out the ten lowest planes in the *z*-coordinate to remove the NaNs in the data that correspond to land. The dataset *asteroid* uses the first ten timesteps from the field "v02" from a deep water asteroid impact simulation [PG18].

Fig. 7 shows a gallery of compression results on our approach, which we refer to as *neurcomp*. We also compare our method to the state-of-the-art compression technique *thresh* [BRLP19], a method for compressing arbitrary tensor data. We limit our comparisons to only *thresh* as Ballester et al. have already demonstrated superior results to a wide variety of existing compression methods. Unless otherwise stated, we use OSPRay [WJA\*16] for rendering and iso-surfacing for visual comparisons.

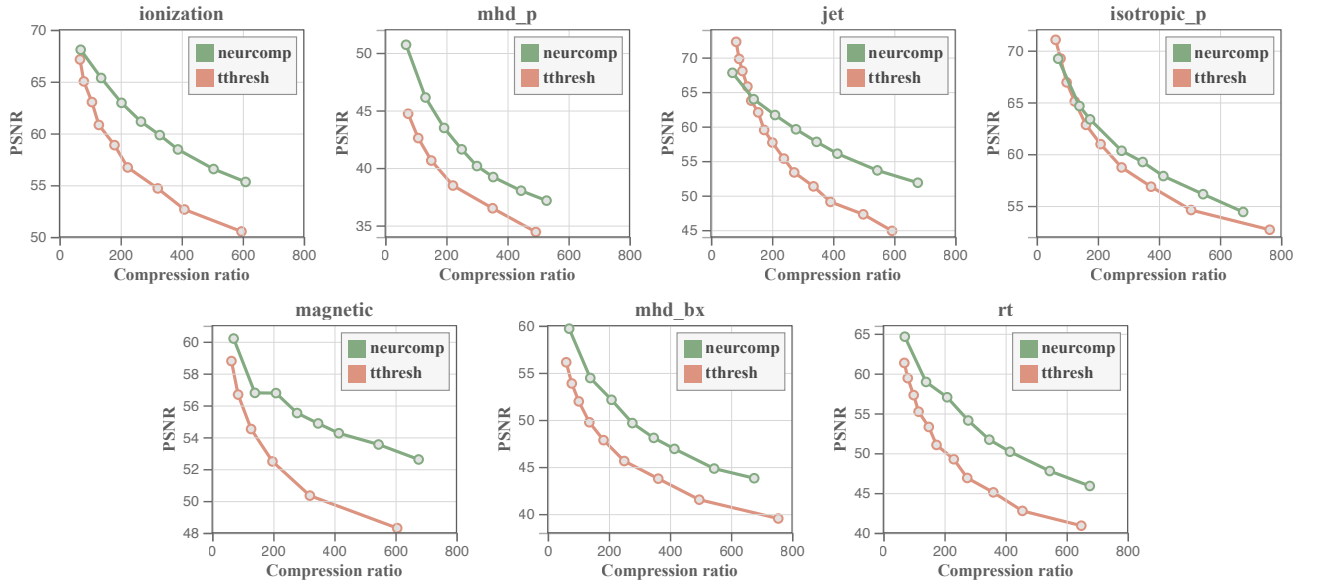
**Implementation Details.** We use Adam [KB14] to optimize our networks, where we found the best results by setting the learning rate to be inversely proportional to the number of weights of the network. Specifically, the learning rate is set as a linear function, varying from  $2 \cdot 10^{-5}$  for 5M parameters, to  $10^{-4}$  for 800K parameters. We ensure that our network makes a prescribed number of passes over the entire volume, where we found 75 iterations to be more than enough for convergence, decaying the learning rate by a factor of 5 every 20 iterations. At each iteration we set the batch size to fill up the GPU memory, in practice ranging from 16K to 64K grid points from the volume sampled uniformly at random.

### 4.1. 3D Scalar Fields

We first compare our method with *thresh* for 3D scalar fields. We run our method over a sequence of compression levels, and plot the



**Figure 7:** Compression results for our approach, *neurcomp*. Left-to-right: *asteroid* (timestep 0), *ionization*, *isotropic\_p*, *magnetic*, and *rt*. Top row: original volumes. Middle row: medium compression. Bottom row: high levels of compression.



**Figure 8:** A comparison of our method with *tthresh* [BRLP19] for 3D scalar fields, plotting compression ratio against PSNR.



**Table 2:** Name, resolution, precision in bits, and file size in MB for the datasets in our experiments. For time-varying datasets *asteroid* and *isabel*, the 4-th dimension of the resolution is the number of timesteps.

Dataset	Resolution	Precision	Size (MB)
mhd_p	$255 \times 255 \times 255$	float32	63
ionization	$248 \times 248 \times 600$	float32	141
jet	$512 \times 336 \times 768$	float32	504
mhd_bx	$512 \times 512 \times 512$	float32	512
isotropic_p	$512 \times 512 \times 512$	float32	512
magnetic	$512 \times 512 \times 512$	float32	512
rt	$512 \times 512 \times 512$	float32	512
asteroid	$300 \times 300 \times 300 \times 10$	float32	1030
isabel	$500 \times 500 \times 90 \times 12$	float32	1030

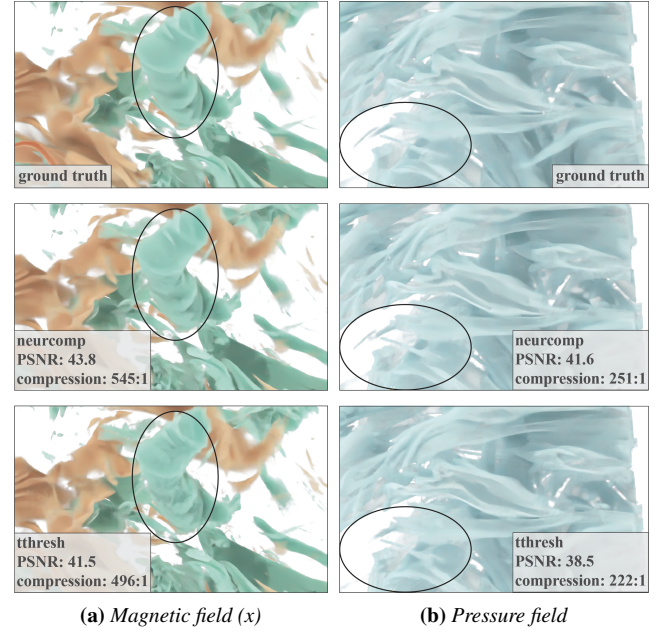
resulting compression ratio against PSNR. For these experiments we do not use gradient regularization, but rather, delegate this to Section 4.4. As *tthresh* accepts an error/accuracy tolerance (e.g. PSNR) as input parameter, rather than compression ratio, we run *tthresh* for a sequence of PSNR values such that they give compression ratios that are roughly in the range of what we consider.

Fig. 8 shows the quantitative results. We find that our method is, largely, an improvement over *tthresh*, with the only exception being the jet volume for low compression ratios. In particular, we find that our method obtains larger gains in performance the higher the compression ratio. This suggests that when our networks are underparameterized (e.g. much fewer parameters than the resolution of the data), they remain robust, and can still obtain good approximations to the given scalar field.

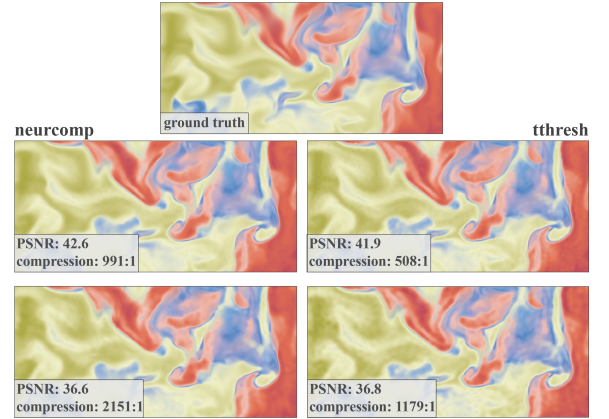
To further qualitatively assess this robustness in high compression regimes, we visually compare volume renderings of our method with *tthresh* for the magneto-hydrodynamic simulation, namely its magnetic field (*mhd\_bx*) and pressure field (*mhd\_p*) in Fig. 9. We find that our method is able to reproduce smoother surfaces in the rendering, as shown in the magnetic field plot (a), while in the pressure field (b), we find that components of the volume, corresponding to the chosen transfer function, are better retained through our method. Indeed, we find that our method gracefully degrades for extreme compression ratios, unlike block-based methods [Lin14, DC16] that produce visible artifacts at block boundaries. Global decomposition-based methods [BRP16, BRLP19] also do not suffer from locality issues, but we find that they can produce high-frequency noise in high compression regimes. We demonstrate this in Fig. 10 for the *rt* dataset, comparing our method with *tthresh* under approximately the same PSNR values. We first note that our method produces compressed representations approximately half the size of *tthresh*. Further, we find that our method produces smoother scalar fields, more reflective of the ground truth as shown on top, unlike the high frequencies reproduced by *tthresh*.

## 4.2. Time-varying Scalar Fields

We evaluate our approach’s capability to compress time-varying data by comparing our method with *tthresh* on *isabel* and *asteroid*,



**Figure 9:** We show visual comparisons between our method and *tthresh* for the magneto-hydrodynamic simulation. For high compression ratios, we find that our method produces smoother results (a) and can better retain features in the volume (b).

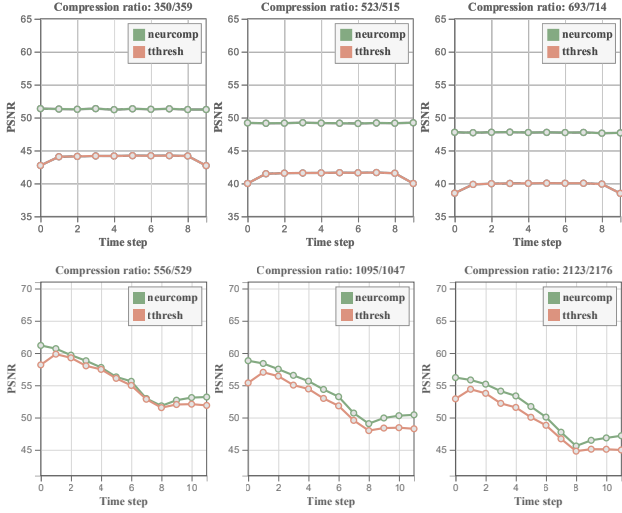


**Figure 10:** Evaluation of our method under extreme compression ratios for the *rt* dataset. We find our method gracefully degrades in performance, whereas *tthresh* tends to reproduce high frequencies.

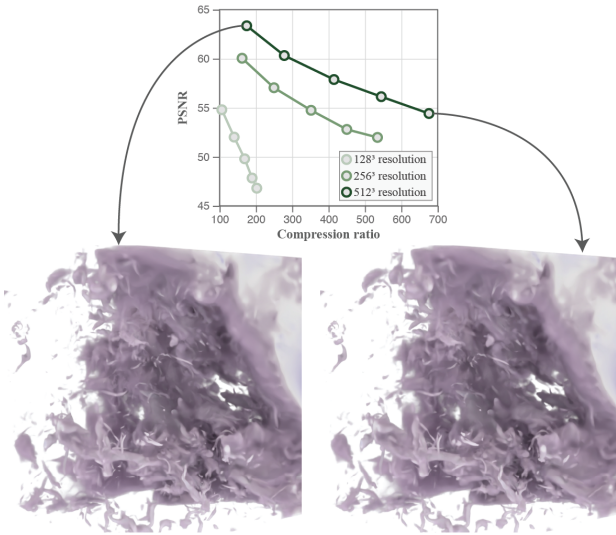
assembling the first twelve time steps from *isabel* and ten consecutive time steps towards the beginning of the *asteroid* simulation. While we show results per time-step, we note that both methods treat the data as 4D volumes.

Fig. 11 show comparisons between our method and *tthresh* over multiple timesteps with regard to 3 different compression ratios. Note that the compression ratios for our method and *tthresh* are slightly different, since it is difficult to precisely set the compression ratio for *tthresh*. As the top of Fig. 11 shows, our approach





**Figure 11:** Comparison against *tthresh* of our approach's performance on the time-varying datasets. Top row: *asteroid*, bottom row: *isabel*. From left-to-right we vary compression ratios, denoting them as  $X/Y$  where  $X$  is the compression ratio of *neurcomp* and  $Y$  is the compression ratio of *tthresh*.

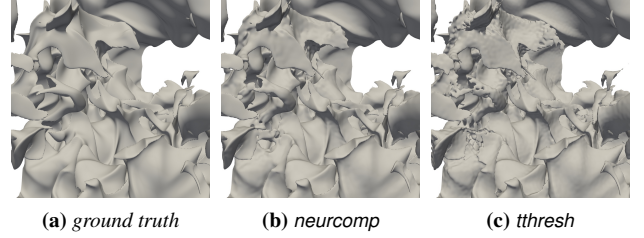


**Figure 12:** We assess the effect of spatial resolution on performance, evaluating our method on spatial crops of different size for the  $512^3$  *isotropic\_p* volume.

achieves superior results on *asteroid* dataset when compared with *tthresh*, with more stability on both ends. In part, this is due to the fine resolution of sampling in time, where the simulation evolves slowly. For the *isabel* dataset, our method performs comparable to *tthresh* at low compression ratio, except for the start and end timesteps where we outperform, as shown in the bottom of Fig. 11. For higher compression levels, however, our method gains larger boosts in performance over *tthresh* at all timesteps, showing the preference of our method for high-compression ratio scenarios.

**Table 3:** A comparison of *tthresh* to our gradient-regularized network.

Method	Compression	PSNR	Gradient PSNR
neurcomp	545:1	47.9	50.7
tthresh	258:1	47.6	46.6



**Figure 13:** We show isosurfaces for the *rt* dataset with our gradient-regularized network and *tthresh*. Though both have roughly the same PSNR, our method suppresses high-frequency details.

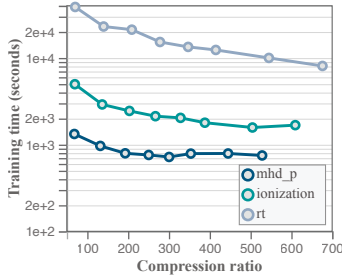
### 4.3. Spatial Resolution

Here we examine the impact of spatial resolution on the quality of our compression method. In particular, we wish to gain insight on the following question: do coordinate-based MLPs benefit from high spatial resolution? To test this, we took the *isotropic\_p*  $512^3$  volume, and grabbed centered crops of size  $256^3$  and  $128^3$ . We expect this volume to contain similar statistics for sufficiently large spatial crops, since it corresponds to isotropic turbulence at relatively small scales. We then ran our method on the volumes, setting the network size-to-volume ratio ( $\frac{C}{m}$ ) constant, namely 50, 100, 150, 200, and quantize the network weights to 9 bits. Note that for larger volumes, the storage of the cluster centers leads to overall higher compression ratios, thus for equivalent  $\frac{C}{m}$  values lower resolution volumes are given the benefit of the doubt.

Fig. 12 (top) shows the results, plotting performance as PSNR with respect to ground truth. We can see that the higher the spatial resolution, the better the performance, despite (a) the network sizes setup in the same manner, and (b) the higher compression ratios for the larger volumes. We view this as an encouraging property, namely that network capacity is not strictly tied with the spatial resolution. On the bottom of the figure we show volume renderings from low and high-end compression ratios, illustrating that our approach captures the predominant features of the volume even for high levels of compression.

### 4.4. Gradient Regularization

Last, we show how gradient regularization can be used to produce better-behaved scalar fields in comparison to prior works. Specifically, for the *rt* dataset we obtain target gradients through central finite differencing, and train our method using the loss of Eq. 3. We provide our resulting PSNR as a target accuracy for *tthresh*, yielding comparable PSNR values as shown in Table 3. However, a gap exists between the methods in terms of the Gradient-based PSNR. As discussed in Sec. 3.1.2, the effect of gradient regularization will



**Figure 14:** We plot training times for three scalar fields, as a function of compression ratio.

be prominent when isosurfacing the volume, and we can observe in Fig. 13 that  $t_{\text{thresh}}$  indeed produces isosurfaces that contain considerable noise – this corroborates findings in Fig. 9(a) and Fig. 10. Indeed, the preservation of higher-order information is nontrivial through existing compression methods, whereas our method can incorporate gradient preservation as part of optimization.

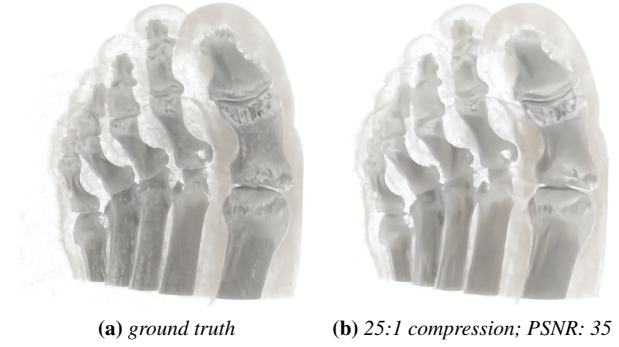
## 5. Discussion

Although our method can achieve state-of-the-art results for compressing scalar fields, we acknowledge several limitations. The primary drawback is the high computation time for training. In Fig. 14 we plot training times, as a function of compression ratio, for three scalar fields of different spatial resolution – we observe similar times for different volumes with equivalent resolution. These times were computed on a single node using an NVIDIA V100 GPU with 16GB of memory. We foresee our method as being suitable for large simulations running on supercomputers, where as the simulation completes one could run a deep learning job for compression.

We also note that the training times are rather pessimistic – we find that our network converges to comparable PSNR values in, roughly,  $\frac{2}{3}$  the reported times, and so there are opportunities to speed up training. However, as shown, for large volumes training can be quite slow (requiring a few hours per volume). Interestingly, and perhaps counter-intuitively, as the compression ratio increases it takes less time to compress the data, due to the fact that higher compression ratios utilize smaller network architectures. So, unlike many standard techniques that start from a lossless perspective and work harder to compress, we control this at the architecture level.

We will address this limitation in future work by investigating multiresolution methods for training [JJH20], adapting our network to take spatial locality into account. A major benefit of our approach to compression is that we do not need to store the entire volume in memory at once, since during training we need only access random samples of the field. Addressing limitations in the time required to train will enable us to realize this benefit.

Another limitation related to high computation time is the lack of generalization: the function that is learned is specific to a single volume. We argue that tailoring a neural network to a single volume is precisely what gives us such good performance, as similarly demonstrated in recent work within shape representations [DNJ20]. Nevertheless, we have also shown the efficacy of our method for



**Figure 15:** We show our method's performance for a medical image, here in the case of foot ( $256^3$ , 8-bit precision).

time-varying scalar fields, and so for future work, we plan to incorporate other factors common to simulation-based data, e.g. data composed of multiple fields and associated with simulation parameters, within our network design. Indeed the notion of generalization is limited for volumetric data within the visualization community, often restricted to fields/time steps produced from a single simulation [HW19, HZX\*20] or fixing these parameters and instead considering multiple simulation parameters [HWG\*19]. Our approach is general enough to accommodate these factors.

The use of neural networks for representing volumetric data places some limitations on the types of volumes that we can support. In particular, our method faces limitations in handling noisy volumes, e.g. those produced from medical imaging. The main issue is not in capturing the predominant features of the volume – our method performs well on this matter, please see Fig. 15 for an example on the foot volume. Rather, the use of neural networks makes it challenging to preserve the noise itself. This is also problematic for gradient regularization, where it is challenging to extract useful gradient signal for our network if noise is present. On the other hand, we demonstrated that simple finite difference schemes for simulation data proves helpful for regularization, and we expect higher-order numerical schemes for derivative estimation should also prove useful. Beyond gradients, we can also regularize the learned function with *any* type of differential expression, and this can lead the way to preserving the physics associated with a given simulation as part of our compression approach.

Despite the limitations, we are optimistic about the use of neural networks for scientific data compression. Besides the clear advantages shown over other architectures [SMB\*20], coordinate-based MLPs offer significant flexibility in designing constraints for data preservation. Future work may show that such networks can better preserve a variety of features in volumetric data.

## Acknowledgements

This work is supported in part by the National Science Foundation (NSF) under grant numbers IIS-2007444 and IIS-2006710, and by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, under Award Number(s) DE-SC-0019039.

## References

- [BLL18] BERGER M., LI J., LEVINE J. A.: A generative model for volume rendering. *IEEE transactions on visualization and computer graphics* 25, 4 (2018), 1636–1650. 2
- [BRLP19] BALLESTER-RIPOLL R., LINDSTROM P., PAJAROLA R.: TTHRESH: Tensor compression for multidimensional visual data. *IEEE transactions on visualization and computer graphics* (2019). 2, 6, 7, 8
- [BRP16] BALLESTER-RIPOLL R., PAJAROLA R.: Lossy volume compression using Tucker truncation and thresholding. *The Visual Computer* 32, 11 (2016), 1433–1446. 1, 2, 8
- [CCM04] COOK A. W., CABOT W., MILLER P. L.: The mixing transition in Rayleigh-Taylor instability. *Journal of Fluid Mechanics* 511 (2004), 333–362. doi:10.1017/S0022112004009681. 6
- [CZ19] CHEN Z., ZHANG H.: Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 5939–5948. 3
- [DC16] DI S., CAPPELLO F.: Fast error-bounded lossy HPC data compression with SZ. In *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)* (2016), IEEE, pp. 730–739. 2, 8
- [DMG20] DÍAZ J., MARTON F., GOBBETTI E.: Interactive spatio-temporal exploration of massive time-varying rectilinear scalar volumes based on a variable bit-rate sparse representation over learned dictionaries. *Computers & Graphics* (2020). 2
- [DNJ20] DAVIES T., NOWROUZEZAHRAI D., JACOBSON A.: Overfit neural networks as a compact shape representation. *arXiv preprint arXiv:2009.09808* (2020). 3, 10
- [GBG\*14] GYULASSY A., BREMER P.-T., GROUT R., KOLLA H., CHEN J., PASCUCCI V.: Stability of dissipation elements: A case study in combustion. *Computer Graphics Forum* 33, 3 (2014), 51–60. 6
- [GIGM12] GOBBETTI E., IGLESIAS GUTIÁN J. A., MARTON F.: COVRA: A compression-domain output-sensitive volume rendering architecture based on a sparse representation of voxel blocks. *Computer Graphics Forum* 31, 3pt4 (2012), 1315–1324. 2
- [GLDL14] GUO F., LI H., DAUGHTON W., LIU Y.-H.: Formation of hard power laws in the energetic particle spectra resulting from relativistic magnetic reconnection. *Phys. Rev. Lett.* 113 (Oct. 2014), 155005. doi:10.1103/PhysRevLett.113.155005. 6
- [GWGS02] GUTHE S., WAND M., GONSER J., STRASSER W.: Interactive rendering of large volume data sets. In *IEEE Visualization, 2002. VIS 2002.* (2002), IEEE, pp. 53–60. 2
- [GYH\*20] GUO L., YE S., HAN J., ZHENG H., GAO H., CHEN D. Z., WANG J.-X., WANG C.: SSR-VFD: Spatial super-resolution for vector field data analysis and visualization. In *2020 IEEE Pacific Visualization Symposium (PacificVis)* (2020), IEEE Computer Society, pp. 71–80. 3
- [HAESB20] HAO Z., AVERBUCH-ELOR H., SNAVELY N., BELONGIE S.: DualSDF: Semantic shape manipulation using a two-level representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 7631–7641. 3
- [HMD15] HAN S., MAO H., DALLY W. J.: Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. *arXiv preprint arXiv:1510.00149* (2015). 5
- [HSB\*20] HOANG D., SUMMA B., BHATIA H., LINDSTROM P., KLACANSKY P., USHER W., BREMER P.-T., PASCUCCI V.: Efficient and flexible hierarchical data layouts for a unified encoding of scalar field precision and resolution. *IEEE Transactions on Visualization and Computer Graphics* (2020). 2
- [HTZ\*19] HAN J., TAO J., ZHENG H., GUO H., CHEN D. Z., WANG C.: Flow field reduction via reconstructing vector data from 3-d streamlines using deep learning. *IEEE computer graphics and applications* 39, 4 (2019), 54–67. 6
- [HW19] HAN J., WANG C.: TSR-TVD: Temporal super-resolution for time-varying data analysis and visualization. *IEEE Transactions on Visualization and Computer Graphics* 26, 1 (2019), 205–215. 3, 10
- [HW20] HAN J., WANG C.: SSR-TVD: Spatial super-resolution for time-varying data analysis and visualization. *IEEE Transactions on Visualization and Computer Graphics* (2020). 3
- [HWG\*19] HE W., WANG J., GUO H., WANG K.-C., SHEN H.-W., RAJ M., NASHED Y. S., PETERKA T.: InSituNet deep image synthesis for parameter space exploration of ensemble simulations. *IEEE transactions on visualization and computer graphics* 26, 1 (2019), 23–33. 2, 10
- [HZRS16] HE K., ZHANG X., REN S., SUN J.: Identity mappings in deep residual networks. In *European conference on computer vision* (2016), Springer, pp. 630–645. 4
- [HZX\*20] HAN J., ZHENG H., XING Y., CHEN D. Z., WANG C.: V2V: A deep learning approach to variable-to-variable selection and translation for multivariate time-varying data. *IEEE Transactions on Visualization and Computer Graphics* (2020). 2, 10
- [IKK12] IVERSON J., KAMATH C., KARYPIS G.: Fast and effective lossy compression algorithms for scientific datasets. In *European Conference on Parallel Processing* (2012), Springer, pp. 843–856. 2
- [IP99] IHM I., PARK S.: Wavelet-based 3D compression scheme for interactive visualization of very large volume data. *Computer graphics forum* 18, 1 (1999), 3–15. 2
- [IS15] IOFFE S., SZEGEDY C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning* (2015), pp. 448–456. 4
- [JGG20] JAKOB J., GROSS M., GÜNTHER T.: A fluid flow data set for machine learning and its application to neural flow map interpolation. *IEEE Transactions on Visualization and Computer Graphics* (2020). 2
- [JJHZ20] JIANG Y., JI D., HAN Z., ZWICKER M.: Sdfdiff: Differentiable rendering of signed distance fields for 3d shape optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 1251–1261. 10
- [KB14] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014). 6
- [LDT\*18] LIANG X., DI S., TAO D., LI S., LI S., GUO H., CHEN Z., CAPPELLO F.: Error-controlled lossy compression optimized for high compression ratios of scientific datasets. In *2018 IEEE International Conference on Big Data (Big Data)* (2018), IEEE, pp. 438–447. 2
- [Lin14] LINDSTROM P.: Fixed-rate compressed floating-point arrays. *IEEE transactions on visualization and computer graphics* 20, 12 (2014), 2674–2683. 2, 8
- [LPW\*08] LI Y., PERLMAN E., WAN M., YANG Y., MENEVEAU C., BURNS R., CHEN S., SZALAY A., EYINK G.: A public turbulence database cluster and applications to study lagrangian evolution of velocity increments in turbulence. *Journal of Turbulence*, 9 (2008), N31. 6
- [LSE\*11] LAKSHMINARASIMHAN S., SHAH N., ETHIER S., KLASKY S., LATHAM R., ROSS R., SAMATOVA N. F.: Compressing the incompressible with ISABELA: In-situ reduction of spatio-temporal data. In *European Conference on Parallel Processing* (2011), Springer, pp. 366–379. 2
- [MON\*19] MESCHEDER L., OECHSLE M., NIEMEYER M., NOWOZIN S., GEIGER A.: Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 4460–4470. 2, 3
- [MST\*20] MILDENHALL B., SRINIVASAN P. P., TANCIK M., BARRON J. T., RAMAMOORTHY R., NG R.: NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV* (2020). 2, 3
- [Mur93] MURAKI S.: Volume data and wavelet transforms. *IEEE Computer Graphics and applications* 13, 4 (1993), 50–56. 2
- [OMN\*19] OECHSLE M., MESCHEDER L., NIEMEYER M., STRAUSS T., GEIGER A.: Texture fields: Learning texture representations in function space. In *Proceedings of the IEEE International Conference on Computer Vision* (2019), pp. 4531–4540. 3

- [PFS\*19] PARK J. J., FLORENCE P., STRAUB J., NEWCOMBE R., LOVEGROVE S.: DeepSDF: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 165–174. [2](#), [3](#)
- [PG18] PATCHETT J. M., GISLER G. R.: Deep water impact ensemble data set in 2018 ieee scivis contest, 2018. [6](#)
- [PNG\*18] PETERKA T., NASHED Y. S., GRINDEANU I., MAHADEVAN V. S., YEH R., TRICOCHÉ X.: Foundations of multivariate functional approximation for scientific data. In *2018 IEEE 8th Symposium on Large Data Analysis and Visualization (LDAV)* (2018), IEEE, pp. 61–71. [2](#)
- [RGG\*13] RODRIGUEZ M. B., GOBBETTI E., GUITIÁN J. A. I., MAKHINYA M., MARTON F., PAJAROLA R., SUTER S. K.: A survey of compressed GPU-based direct volume rendering. In *Eurographics (STARs)* (2013), pp. 117–136. [2](#)
- [SMB\*20] SITZMANN V., MARTEL J., BERGMAN A., LINDELL D., WETZSTEIN G.: Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems* 33 (2020). [2](#), [3](#), [4](#), [10](#)
- [SMP13] SUTER S. K., MAKHYNIA M., PAJAROLA R.: TAMRESH - tensor approximation multiresolution hierarchy for interactive volume visualization. *Computer Graphics Forum* 32, 3pt2 (2013), 151–160. [1](#), [2](#)
- [SPCT18] SOLER M., PLAINCHAULT M., CONCHE B., TIERNY J.: Topologically controlled lossy compression. In *2018 IEEE Pacific Visualization Symposium (PacificVis)* (2018), IEEE, pp. 46–55. [2](#)
- [SW03] SCHNEIDER J., WESTERMANN R.: Compression domain volume rendering. In *IEEE Visualization, 2003. VIS 2003.* (2003), IEEE, pp. 293–300. [2](#)
- [TDCC17] TAO D., DI S., CHEN Z., CAPPELLO F.: Significantly improving lossy compression for scientific data sets based on multidimensional prediction and error-controlled quantization. In *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)* (2017), IEEE, pp. 1129–1139. [2](#)
- [TSM\*20] TANCIK M., SRINIVASAN P., MILDENHALL B., FRIDOVICH-KEIL S., RAGHAVAN N., SINGHAL U., RAMAMOORTHY R., BARRON J., NG R.: Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems* 33 (2020). [3](#)
- [WBK04] WANG W., BRUYERE C., KUO B.: Competition data set and description in 2004 ieee visualization design contest, 2004. [6](#)
- [WCTW19] WEISS S., CHU M., THUEREY N., WESTERMANN R.: Volumetric isosurface rendering with deep learning-based super-resolution. *arXiv preprint arXiv:1906.06520* (2019). [3](#)
- [WJA\*16] WALD I., JOHNSON G. P., AMSTUTZ J., BROWNEE C., KNOLL A., JEFFERS J., GÜNTHER J., NAVRÁTIL P.: Ospray-a cpu ray tracing framework for scientific visualization. *IEEE transactions on visualization and computer graphics* 23, 1 (2016), 931–940. [6](#)
- [WMB\*11] WOODRING J., MNISZEWSKI S., BRISLAWN C., DEMARLE D., AHRENS J.: Revisiting wavelet compression for large-scale climate data using JPEG 2000 and ensuring data precision. In *2011 IEEE Symposium on Large Data Analysis and Visualization* (2011), IEEE, pp. 31–38. [1](#), [2](#)
- [WN] WHALEN D., NORMAN M. L.: Competition data set and description. in 2008 ieee visualization design contest (2008). [6](#)
- [XFCT18] XIE Y., FRANZ E., CHU M., THUEREY N.: tempoGAN: A temporally coherent, volumetric GAN for super-resolution fluid flow. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–15. [3](#)
- [YL95] YEO B.-L., LIU B.: Volume rendering of DCT-based compressed 3D scalar data. *IEEE Transactions on Visualization and Computer Graphics* 1, 1 (1995), 29–43. [1](#), [2](#)