Overfit Neural Networks as a Compact Shape Representation

Thomas Davies [1], Derek Nowrouzezahrai [2] and Alec Jacobson[1]

[1]University of Toronto, Canada [2]McGill University, Canada

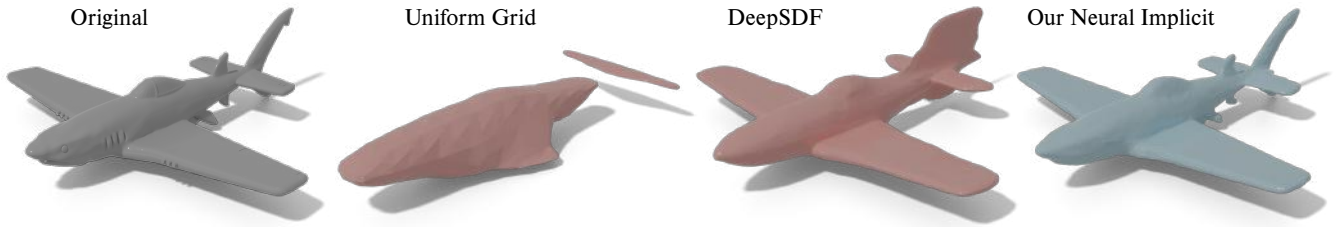

Figure 1: Geometries can be represented with infinite resolution by their continuous signed distance fields (SDFs). The SDF of a shape can be approximated by storing values on a regular grid. Uniform grids are wasteful for storing values far from the surface, resulting in poor reconstructions for small grid sizes. Comparably, a DeepSDF [PFS*19] model can more effectively encode the original shape but requires the shape is consistently aligned and semantically similar (airplanes, cars, boats, etc.) to shapes within the training set, and fails to capture the unique characteristics of the model (i.e. shark fins). Our *Neural Implicit* format is produced by overfitting a neural network to the single geometry directly, providing a compact representation with far greater accuracy, regardless of class or orientation, compared to uniform grids of the same memory impact (64 kB shown here). gpvillamil (right) under CC BY.

Abstract

Neural networks have proven to be effective approximators of signed distance fields (SDFs) for solid 3D objects. While prior work has focused on the generalization power of such approximations, we instead explore their suitability as a compact – if purposefully overfit – SDF representation of individual shapes. Specifically, we ask whether neural networks can serve as first-class implicit shape representations in computer graphics. We call such overfit networks Neural Implicits. Similar to SDFs stored on a regular grid, Neural Implicits have fixed storage profiles and memory layout, but afford far greater accuracy. At equal storage cost, Neural Implicits consistently match or exceed the accuracy of irregularly-sampled triangle meshes. We achieve this with a combination of a novel loss function, sampling strategy and supervision protocol designed to facilitate robust shape overfitting. We demonstrate the flexibility of our representation on a variety of standard rendering and modeling tasks.

1. Introduction

Signed distance fields (SDFs) are a versatile implicit surface representation, useful throughout computer graphics [BBB*97]. Complex objects represented as SDFs can be authored *semi-analytically* by (incrementally) composing geometric primitives with space warping, blending operations, and replicating functions (see inset by Inigo Quilez). However, storing an SDF as a long composition of expressions does not scale, especially to shapes with a high level of (non-procedural) detail. What is the best way to store an SDF?

Approximating an SDF by storing values on a regular grid speeds up evaluation at the cost of precomputation (effectively treating the SDF as 3D table look up). Shapes stored as SDFs on grids benefit by having fixed storage profiles and memory layouts. Unfortunately, this comes at a cost, as grids wastefully store a dense sampling of the SDF far from the surface where the value is smooth and predictable. While octrees and truncated signed distance functions can store SDFs asymptotically more efficiently, their representation incurs non-uniform computational and memory costs (e.g., the octree leaf nodes of one surface are different from those of another).

Meanwhile, explicit representations are ubiquitous as a data format for distributing 3D models: hundreds of millions of meshes are available online. While easier to animate and texture, explicit representations like meshes are cumbersome for shape modeling and querying tasks common in computer vision, simulation and geometric learning. This raises the question: how do we convert mesh assets into implicit representations? Embedding a mesh in,

e.g., a spatial hierarchy to compute point-mesh signed distances is inefficient compared to flat table lookups or evaluation of analytic expressions. Bounding hierarchy distance queries also require divergent computations, e.g., different queries on the same shape can have drastically different computational and memory access costs. Converting to grids or octrees does not avoid their limitations.

We show that *overfitting* a deep neural network to the SDF of a single solid is effective, and we advocate for its consideration as a first-class implicit representation. We show that these overfit neural networks – which we call *Neural Implicits* – are a shape representation that inherit the effectively infinite resolution of implicits but with the computational efficiency of coarse meshes and the memory access uniformity of a fixed grid.

1.1. Problem Context

Implicit representations are especially attractive for geometric machine learning. Voxel occupancy in a 3D image (grid storing inside/outside values per cell) is a homogeneous representation especially amenable to classification and convolution networks [MS15; WSK*15]. Large datasets of 3D meshes (e.g., ShapeNet [CFG*15]) can easily be converted to a voxel grid or grid-based SDF [NLBY18; HSG18; LYF17], so that the network architecture can input a homogenous image format similar to 2D convolution networks. Attempting to learn directly on 3D meshes has been attempted, but architectures become esoteric [HHF*19; WKBS19] and may rely on supplemental handcrafted features (see longer discussion in [BBL*17]). Conversion to point clouds (an unordered set) sidesteps the homogeneity issue by removing dependence on ordering or explicit/implicit knowledge of the shape's manifold structure entirely [QSMG16]. While most of these representations have proven success at classification and recognition tasks, a much more daunting task is generative modeling. Networks that output an entire occupancy grid [MS15; WWX*17] or even a sparse grid [WSLT18] are ultimately limited to small grids incapable of representing fine detail.

In just the past year, there has been an explosion of work sparked by the groundbreaking success of DeepSDF [PFS*19]. Park, Florence, Straub, et al. [PFS*19] approximate the signed distance to a surface as an evaluation of a deep neural network: like any SDF, the input is a query point in space and the output is a signed distance value at that point. Their goal is to learn a latent space for large dataset of class specific shapes. Their network architecture includes a latent code optmized for each shape while network is trained over the whole dataset. This functional representation has been shown to be powerful for generative modeling [MON*19], shape interpolation [LW19], differentiable rendering [LZP*19; NMOG19; JJHZ19], and surface reconstruction [AL19]. In most cases, networks are trained over a large class of shapes with a latent vector to encode each shape in question. The resulting shapes are impressive from the point of view of generative modeling, but inevitably suffer accuracy reproducing any given shape (Figure 1) in the pursuit of generalization. Prior methods have focused on learning class-priors across large datasets to achieve strong results; unfortunately most geometries in the wild are not consistently aligned (Figure 11) and do not belong to some easily discerned shape family. The original DeepSDF briefly considers but quickly discards the idea of overfitting its neural network to each shape individually:

> "Training a specific neural network for each shape is neither feasible nor very useful." — Park, Florence, Straub, et al. [PFS*19]

We propose training a specific neural network for each shape and will show that this is both feasible and very useful.

1.2. Contributions

We demonstrate that overfit neural networks, or *Neural Implicits*, exhibit an interesting combination of the desirable qualities of a shape representation. Overfitting to a single shape is often treated as a test case before attempting generalization over a larger training set. Indeed, if held to the scrutiny of a shape representation for applications in computer graphics, prior overfit results from deep signed distance fields are lacking. We identify issues with prior strategies for defining the training loss, sampling strategies, and supervision. While many previous methods discuss loss functions and sampling independently, we propose a loss function defined by a continuous spatial integral. We discretize this integral using Monte Carlo approximation resulting in a query probing supervised sampling strategy for training. While prior works also employ stochastic sampling, our integral formulation affords direct application of importance sampling. We propose a simple yet effective subset rejection importance sampling strategy that samples close to the input shape's surface without biases observed in existing methods. For each query sample, we conduct a supervised stochastic descent step to update the network weights. Supervised training requires accurate ground-truth signed distance evaluation. For surfaces with non-manifold edges, self-intersections and open boundaries, signing methods used in prior works can fail to behave robustly, often introducing simplification error (Figure 8) even before training begins. We propose using fast winding numbers [BDS*18] as a signing proxy that is exactly correct for solid geometries (those perfectly represented as the level set of a signed distance field) and gracefully degrades for messy input shapes (Figure 9).

We demonstrate the effectiveness of overfit neural networks as a solid shape representation for a variety of tasks in computer graphics, starting with rendering. We compare the economical storage of *Neural Implicits* to existing formats (Figure 14). Compared to decimated meshes (baseline non uniform memory access), we observe that our fixed memory format has similar surface quality. Compared to SDFs stored on a grid (baseline consistent memory access) we observe far better quality.

2. Method

We introduce OverfitSDF, a neural network architecture trained to overfit to a single shapes signed distance function. Once overfit the learned parameter set θ can be used as an efficient and lightweight representation of the shape. We call this format a *Neural Implicit* . The *Neural Implicit* format of a given shape is the learnt network weights of a OverfitSDF model trained on samples drawn from the shape's signed distance function.

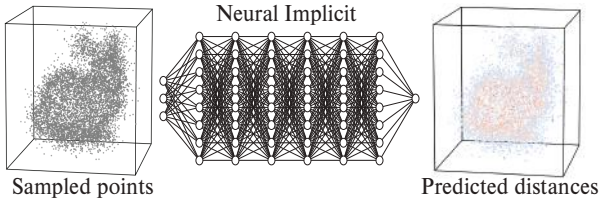

Figure 2: OVERFITSDF network architecture. Given point samples of an object's SDF (left), we train a feed-forward neural network (middle) to predict the signed distances (right) of each input point.

2.1. Overfitting Implicit Fields

A signed distance field is a representation in which, at each point within the field, we can measure the distance from that point to the closest point on any shape within the domain. The sign on the distance field represents the direction to the nearest surface, and indicates whether the point is internal or external to objects in the domain.

The signed distance function (SDF) of a surface can be defined by the metric set Ω of points within the shape, along with metric d.

$$SDF(\mathbf{x}, \delta\Omega) = \begin{cases} -d(\mathbf{x}, \delta\Omega) & x \in \Omega \\ d(\mathbf{x}, \delta\Omega) & x \notin \Omega \end{cases} \quad (1)$$

where $\delta\Omega$ denotes the boundary of metric Ω, and d can be defined as distance from the closest point on $\delta\Omega$ to x.

Our goal is to regress a feed-forward network to approximate the SDF of a given surface ($\delta\Omega$), such that,

$$f_\theta(\mathbf{x}) \approx SDF(\mathbf{x}, \delta\Omega) \quad (2)$$

Once f_θ is overfit to a given shape, the parameter set θ can then be used as a first-class implicit representation of the shape.

2.1.1. Architecture

Our OVERFITSDF (Figure 2) is a feed-forward fully connected network with N layers, of hidden size H. Each hidden layer has ReLU non-linearities, while the output layer is activated by TanH.

Deep neural networks have a tendency to produce their best results when their depth and width are increased. Unfortunately, this increase in complexity proportionally increases memory footprints of our representation and the time required to render. The *Neural Implicit*'s rendered in Figures 1, 7, 10, 11, 12, and 14 all share a common architecture of just 8 fully connected layers with a hidden size of 32 (resulting in just 7553 weights, or 64 kB in memory). Through experimentation on a subset of 1000 mesh geometries from ZHOU and JACOBSON [ZJ16]'s Thingi10k dataset, we find that this configuration yields a good balance between reconstruction accuracy, rendering speed, and memory impact (Figure 3). Our chosen architecture has a 99% reduction in number of parameters and 93% speed up in time to render first frame, while still providing acceptable surface quality when compared to the default architecture in DeepSDF [PFS*19] (without latent optimization).

Our OVERFITSDF network can, in theory [HSW89], learn to emulate any arbitrary topology shape with infinite precision. The network complexity can be increased over our base configuration for smaller surface reconstruction error, or decreased for faster rendering speeds depending on the application. A sample of geometries produced at a number of configurations can be seen in Figure 4.

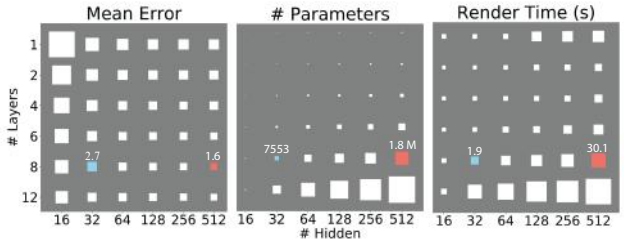

Figure 3: We visualize the role that varying the number of network layers, and hidden layer sizes, play on: (left to right) average reconstruction error, memory footprint and 1-frame render time. Chosen architecture shown in blue, default DeepSDF [PFS*19] architecture (without latent optimization) shown in red.

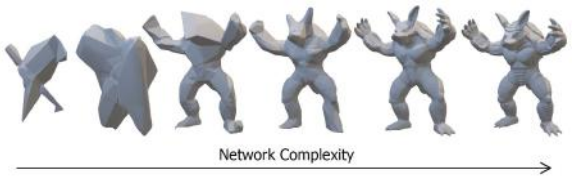

Figure 4: As network complexity of the OVERFITSDF network increases, the models reconstruction quality increases while rendering speed decreases. Depending on the application varying levels of accuracy can be achieved.

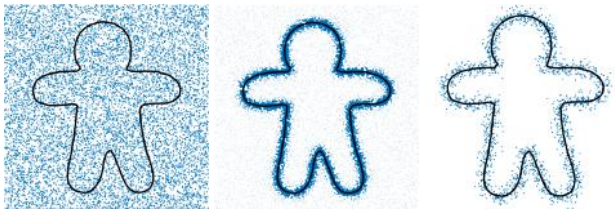

Figure 5: Our importance sampling (right) draws a subset of points from a uniform sample set (left) according to their distance to the surface (middle).

2.1.2. Sampling and Loss

We train each OVERFITSDF on a set $\mathbf{X}$ of points sampled in $\mathbb{R}^3$, along with their corresponding signed distance evaluations, for a given shape. A naïve sampling approach could draw samples uniformly from the bounding sphere. Our setting, however, is unique in that the inside-outside decision boundary of a shape is well represented by the 0-isocontour of the SDF. Learning the signed distance value far from the surface is useful for efficient ray marching (see section 3.1) but in practice we want to focus training on points close to the shape boundary for high quality surface reconstruction. As such, instead of employing a random dart throwing scheme we can pursue a more principled approach that focuses samples on points more "informative" to the boundary transitions (Figure 5).

Several methods focus a network's capacity on points close to a boundary. [LW19] propose a vertex sampling method that draws

samples uniformly from a mesh's vertex positions before perturbing the samples according to an isotropic 3D Gaussian. Sampling based exclusively on vertex positions biases coverage due to the underlying mesh tessellation, leading to unwanted anisotropies due to variations in triangle surface areas, particularly when the Gaussian variance is not carefully set (Figure 6). The state-of-the-art DeepSDF method [PFS*19] improves on this scheme by drawing samples uniformly over triangle surfaces, before similarly perturbing with a Gaussian. Such an approach still introduces a bias due to the non-uniform distribution of nearby mesh faces (Figure 6). Both approaches append an additional set of uniformly sampled spatial points in order to offset their surface bias.

We employ simple least absolute deviations (L1) as our loss function, finding it performs better surface reconstruction when compared to squared error (L2), which is more sensitive to outliers.

$$L_1 = \int_B |SDF(\mathbf{x}) - f_\theta(\mathbf{x})| dx \tag{3}$$

Where f_θ is our OVERFITSDF function, $SDF(x)$ is the true signed distance at $\mathbf{x}$ drawn from bounding volume B. Focusing a network's capacity on specific points of *importance* can be achieved by weighting the loss function to exaggerate error around points of focus.

$$L_{weighted} = \int_B |SDF(\mathbf{x}) - f_\theta(\mathbf{x})| w(\mathbf{x}) dx \tag{4}$$

This method simply scales the loss function with respect to some metric of importance $w(\mathbf{x})$, such that low importance training samples have less affect on the loss. This approach can achieve our goal of biasing to points close to the decision boundary by employing an exponential weighting importance metric on the distance from the surface for any given point.

$$w(\mathbf{x}) = e^{-\beta |SDF(\mathbf{x})|} \tag{5}$$

Where β can be adjusted from 0 for uniform sampling to ∞ for surface point sampling. Unfortunately, in weighting the loss directly, computation is wasted on the forward and backwards passes for points that are far from the shape's surface. For example, a point on the edge of the bounding sphere with a β of 30 will be scaled by $1.92e^{-22}$ having negligible effect on the parameters being optimized, yet the same computational cost as if it did. An effectively equivalent but more efficient approach is to apply the importance metric, $w(\mathbf{x})$, to the sampling of points in bounding volume B instead of scaling the loss.

We propose a simple yet effective subset rejection importance sampling strategy that samples points according to their distance to the input shape's surface (Figure 5). We discretize our continuous loss integral (Eqn. 4) using Monte Carlo approximation resulting in a query probing supervised sampling strategy for training. Given a set, U, of n uniformly sampled points, we aim to subsample m points from U to create a set S such that,

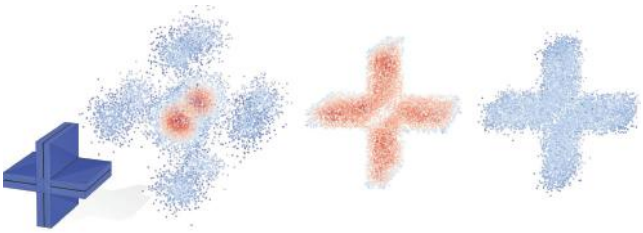

Figure 6: Visualizing sample density of samples when drawing 10^4 points using: (left to right) vertex sampling with $\mathcal{N}(p, 0.1)$ offsets [CZ19], surface sampling with $\mathcal{N}(p, 0.01)$ [PFS*19], and our importance sampling approach with $w = e^{-30|SDF(p)|}$. We color samples according to their density estimated with Gaussian kernel density, normalized by the most dense region from vertex sampling.

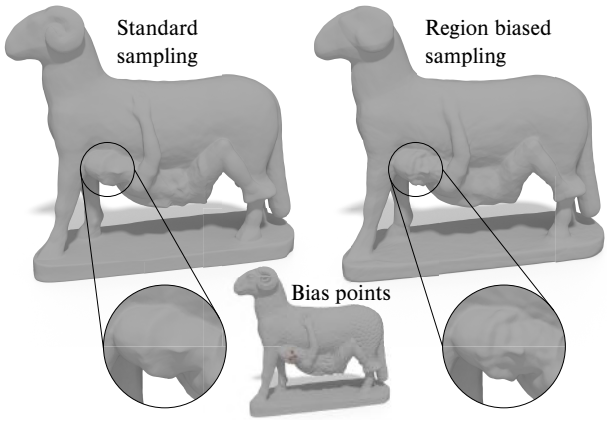

Figure 7: Our importance metric can be additionally weighted by distance from user specified regions. This weighting allows users to specify regions of interest (center; shown in red) yielding improved reconstruction accuracy (right) where desired.

$$\frac{1}{n}\sum_{x \in U} |SDF(\mathbf{x}) - f_\theta(\mathbf{x})| w(\mathbf{x}) \approx \frac{1}{m}\sum_{x \in S} |SDF(\mathbf{x}) - f_\theta(\mathbf{x}|) \tag{6}$$

We find that using Equation 5 as a method of biasing samples close to the surface leads to faster convergence and reduced surface reconstruction error compared to uniform sampling (96 epochs with surface error of 0.00231). While, when compared to DeepSDF's surface sampling scheme we find similar results for both convergence speed and surface quality (average of 86 epochs each, with surface error of 0.00138 and 0.00131 respectively). In practice we generate set U by sampling 10M points within the unit sphere, and employing our importance rejection strategy to populate subset S with 1M points.

A major benefit of our importance sampling scheme is the removal of unintended bias presented by previous approaches, and the complete flexibility in introducing targeted bias. The importance metric, $w(\mathbf{x})$, can be modified to introduce bias towards regions of high curvature, minimum feature size (emulating [PFS*19] bias), or to specific regions of the mesh for areas of high reconstruction importance (Figure 7). This flexibility allows for greater use of

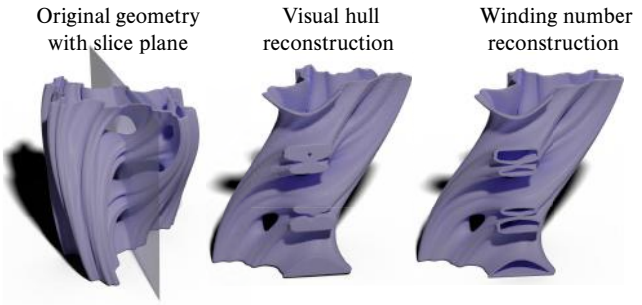

Figure 8: Our approach to signing allows us to support converting non-manifold mesh, without sacrificing the true topology of the mesh. Unlike [PFS*19] visual hull method (middle), our method(right) maintains complex internal structures. Virtox (left) under CC BY.

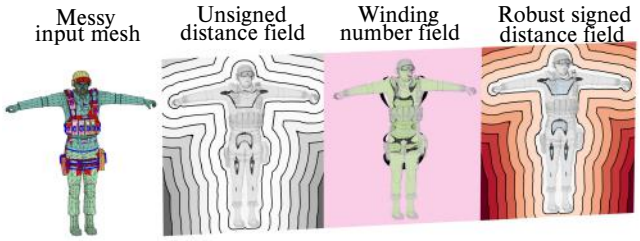

Figure 9: Our conversion process supports non-watertight meshes with open boundaries, self intersections, and non-manifold edges.

the network's capacity on regions or features important to the user, without increasing overall network complexity.

2.2. Signing Distances

The most obvious approach for computing the sign of a point in space is to use the shape's surface normals; calculating the dot product of the normal and the direction vector to point $\mathbf{x}$. If the direction vector points the same way as the surface normal, then $\mathbf{x}$ is external to the surface, and internal otherwise. Unfortunately, this simple process assumes that the input shape is a watertight (closed, non-intersecting, manifold) mesh.

We instead sign our distances with generalized winding numbers [JKS13] enabling us to process meshes with self-intersections, non-manifold pieces, and open boundaries. Previous approaches for signing distances for learning implicit fields either voxelized the space [MON*19], requiring watertight inputs, or a computationally expensive visibility hull [PFS*19], significantly reducing model complexity and "closing" off internal structures (Figure 8) before training even begins. In contrast our method signs distances exactly correct for solid geometries (those perfectly represented as the level set of a signed distance field) and gracefully degrades for messy input shapes (Figure 9).

For fast and efficient sign evaluation we use fast winding numbers [BDS*18], a tree based algorithm for fast approximation of generalized winding numbers. With fast winding numbers we can generate training set $\mathbf{X}$ and overfit our network to even the most problematic meshes (Figure 9), in an average of 90s.

2.3. The *Neural Implicit* File Format

The *Neural Implicit* file format is designed to be simple to consume and integrate into existing pipelines currently relying on classic SDF representations. For each trained OVERFITSDF the chosen network architecture and geometry transformation matrix (since all geometries are normalized to the unit sphere) are written as the first bytes before encoding the network's learnt parameter set θ into our binary format.

Our homogeneous *Neural Implicit* format allows for a single query implementation regardless of network architecture used in the overfitting process. The fixed storage profiles and memory layout of our learnt implicit functions provide consistent query and rendering speeds. Once trained, our *Neural Implicit* format can be treated as any other first class implicit representation.

2.4. Rendering Neural Implicits

Our *Neural Implicit* representation can be treated as its classical counter part (SDF) and rendered efficiently using ray marching. Ray marching [Har96] is a common technique for rendering implicit fields where rays are initialized in the image plane and iteratively "marched" along each ray by a step size equal to the signed distance function value at the point. A single ray is marched until it is sufficiently close to the surface (ε) or has taken the maximum allowable steps along the ray. For more information on ray marching see Morgan McGuire's notes at www.casual-effects.com.

Traditional ray marching can be trivially modified to replace queries against an SDF grid (or equivalent structure) with inferences against our OVERFITSDF. We initialize the starting position of each ray to be it's first intersection with the unit sphere, since all *Neural Implicit*s are normalized to lay within. Rays that do not intersect the unit sphere are pruned from the set before marching begins. As rays of the image converge at different times, we employ a dynamic batching scheme that composes batches of points for inference based on a mask buffer which tracks rays that have converged to the surface or reached the maximum number of steps. Additionally, the dynamic batching method allows us to append additional points when surface normals are required for shading.

3. Implementation and Results

Our *Neural Implicit* representation can be used in applications as if it were the true signed distance field of the shape. Here, we demonstrate OVERFITSDF's ability to learn a shape's SDF function for fast and efficient rendering and as a compressed representation of the original geometry.

We implement OVERFITSDF networks in Tensorflow [MAP*15] while point sampling and mesh processing are implemented in libigl [JPS*16]. We train our model for a maximum of 100 epochs and allow early stopping for geometries that converge quickly. We use the ADAM optimizer [KB14] with a fixed learning rate of 10^{-4}. These settings generalized well across a wide range of geometries (see Fig. 12 and Fig. 15).

Figure 10: *Neural Implicit*s admit many trivial interactive manipulations, the models predicted distance's can be modified through boolean operations similar to any implicit field. See accompanying video for animation.

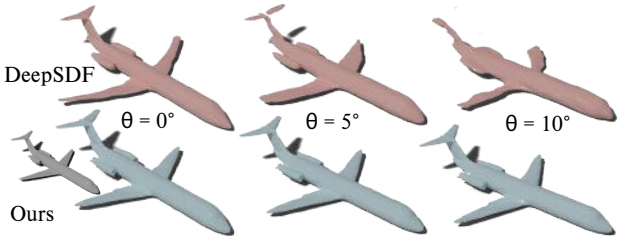

Figure 11: DeepSDF [PFS*19] reconstruction quality degrades quickly for geometries not aligned to default orientation per class. Our method converges to the same quality regardless of orientation. See accompanying video for animation.

3.1. Render Performance

We implement our renderer in CUDA as kernels for sphere marching, fragment shading and batch preparation. We use the CUTLASS [KMDT18] CUDA linear algebra library for fast and efficient strided GEMM (general matrix multiplication) calculation, required for inference against OverfitSDF models.

We achieve an average frame rate of *34 FPS* when rendering at 512 by 512 resolution on an Nvidia P100 GPU across a subset of the Thingi10k dataset. Although not acceptable for real-time rendering applications, this result is a significant improvement over previous learnt implicit rendering pipelines. In [LZP*19] rendering is optimized to 1 FPS by overstepping along all rays by a factor of 50%, increasing the convergence criteria, and implementing a coarse to fine strategy. These optimizations could further improve our rendering speed but would reduce the overall quality of the renders and were not integrated.)

As our representation is a learnt representation of the SDF we inherit all the benefits of traditional implicit functions. Our *Neural Implicit* can be smoothly interpolated in implicit space, and can be interactively modified with constructive solid geometry operations shown in Figure 10.

3.2. Large Scale Conversion of Datasets

Training deep neural networks on large geometric datasets has classically been a cumbersome, time consuming task. For our *Neural Implicit* representation to be effective, any geometry must be able to be converted in a reasonable amount of time. Due to the minimal complexity of our base configuration (8 layers of 32 neurons) we find that we can overfit our model to any geometry in an average of

Figure 12: Thingi10k models [ZJ16] compressed to **64kB** as *Neural Implicits*. Our representation gracefully scales to high-quality reconstructions as its footprint increases, using an order of magnitude less memory than alternative representations at equal quality.

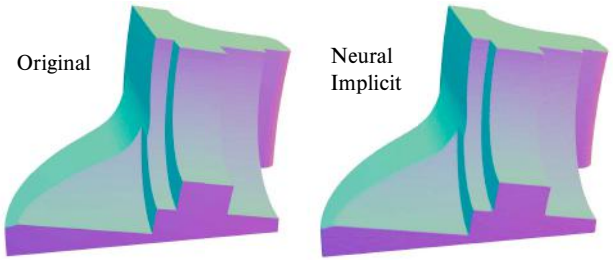

Figure 13: Our *Neural Implicit* format encodes sharp edges with a high degree of accuracy.

90s. Additionally, with only 64kB of memory required, many models can be trained concurrently on modern GPUs without nearing the memory limits. Converting the entirety of the 10,000 models in the Thingi10k dataset [ZJ16] on an Nvidia Titan RTX took just 16 hours on a single GPU or 4 hours when trained in parallel across 4 Nvidia Titan RTX cards.

Conversion of the Thingi10k dataset from mesh format to *Neural Implicit* format reduces the overall storage impact from 38.85 GB to just 640 MB. Comparatively if DeepSDF [PFS*19] could be trained on the same dataset, all geometries would be compressed to just 7 MB. Unfortunately, through our experiments we found that DeepSDF results depend on (a) all geometries belonging to a single class (b) all geometries being consistently oriented and (c) the number of geometries in the dataset. We can see in Figure 11 that rotating the input mesh by just a few degrees results in drastic reconstruction errors, while OverfitSDF produces consistent results regardless of orientation. With this experiment, we demonstrate why DeepSDF failed to converge to reasonable results on the Thingi10k dataset, since geometries are arbitrarily oriented and of no specific class. Additionally, even if DeepSDF did not suffer from the aforementioned problems training on the full 10,000 geometry dataset would be intractable due to the memory required and storing and optimizing the 10,000 shape embeddings (without significantly reducing embedding size, network complexity, or the dataset being converted).

As many of the geometries in Thingi10k dataset are organic "smooth" shapes, we also verify that our method is capable of maintaining sharp edges in reconstructions. We find that our network is able to recreate sharp edges (Fig. 13 & 16) with a high level of accuracy, despite not being specifically biased to do so.

In our conversion of the Thingi10k dataset, the architecture was fixed, yielding an efficient and constant memory representation. However, if surface reconstruction quality is a priority, the focus can instead shift to an error driven surface fitting (similar too classical approaches [OBA*05]), scaling network complexity according to the input geometry. As each OVERFITSDF produced encodes its own architecture, this change will result in simple geometries being encoded into minimal parameter configurations (base: 7553) while topologically complex geometries can be represented by higher resolution configurations. The effect of this error driven optimization approach can be seen in Figure 16, where a simple grid search was performed until surface error met a target goal. Based on our conversion of the Thingi10k dataset, we find that majority of models are represented well by our base configuration (Fig. 15), where the tails of the error distribution could be retrained with additional complexity until some target surface error is achieved.

3.3. Mesh Compression

All images in Figure 12 were rendered from *Neural Implicit* generated from our base configuration, resulting in a total parameter set of 7553 tuned to each shape's implicit function. At just 64 kB of memory we find that our lightweight representation can capture complex topologies with relatively high resolution compared to uniform signed distance grids or decimated mesh if similar memory footprints. For the comparison in Figure 14, each geometry was converted to a *Neural Implicit* in our base configuration of 7553 parameters, and is visualized beside the rendered result of a uniformly sampled SDF grid with 20^3 samples, along with the original mesh adaptively decimated [GH97] down to contain only 7600 floats. Compared to decimated meshes (baseline non-uniform format), we observe that our homogeneous format has similar surface quality. Compared to SDFs stored on a grid (baseline uniform format) we observe far better quality. Additionally, we see that our approach better captures high frequency surface detail than both representations, often producing results that more closely match the "style" of the original shape.

We quantify our methods robustness by converting the entirety of the Thingi10k [ZJ16] dataset to our *Neural Implicit* format, measuring the average surface error (Eqn. 7) and training loss. The training loss reported is the mean of errors between the true and predicted SDF values at points sampled using our importance metric outlined in Section 2.1.2. The surface error is the sum of errors at points along the shape's 0-isocontour,

$$\text{Surface Error} = \frac{1}{N}\sum_{i=1}^{N} |f_\theta(p_i)| \tag{7}$$

We use these simple metrics in conjunction for their ability to not only measure error at the surface but also error within the bounding volume. Errors within the bounding volume manifest as increased rendering times or holes in the model during rendering, while surface errors are clear when marching cubes over the learnt SDF field.

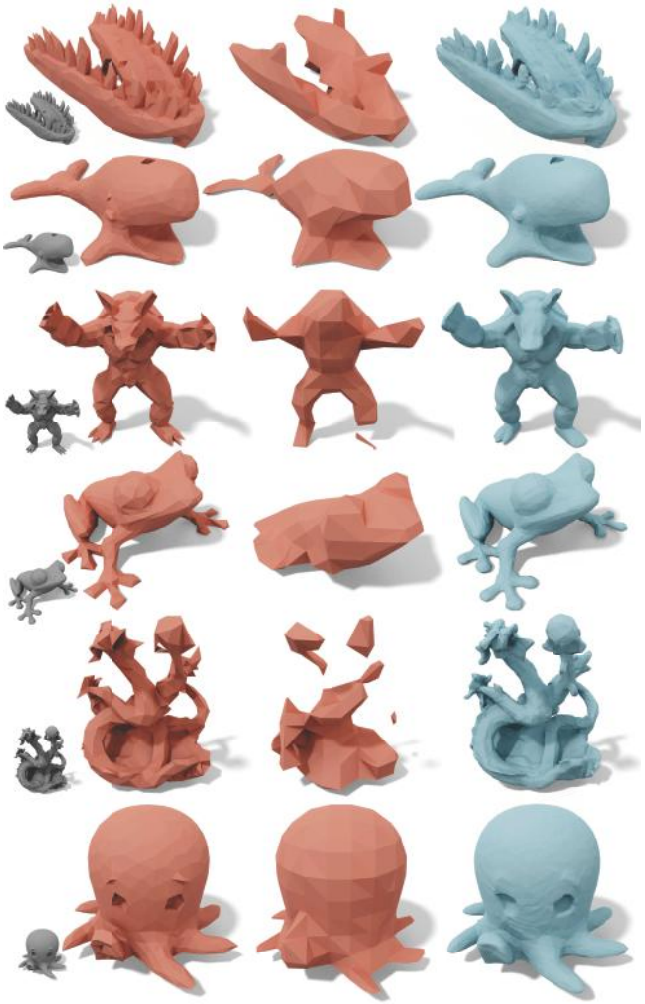

Figure 14: Our learnt *Neural Implicit* format (right) can be shown to better approximate the original surface (grey, inset) compared to adaptive decimation of the original triangle mesh [GH97] (left) and uniform signed distance grid (middle) with equal memory impact. gpvillamil (skull), Makerbot (whale), morenaP (frog), artec3d (dragon), JuliaTruchsess (octopus) under CC BY.

We sample 100,000 surface points for measuring average surface error, and assess converged loss against the training set of $1M$ points. The results on the entirety of the Thingi10k dataset are visualized in Figure 15. We find that at this configuration that 93% of the 10,000 geometries in the diverse Thingi10k dataset converge to a surface error below 0.003, with no model exceeding 0.01 (worst case, of 0.0097 shown in Fig. 16).

4. Limitations and Future Work

In some cases, when a mesh is of exceptional topological , we find that our base configuration of only 7553, simply cannot represent the highly non-linear implicit field. As shown in Figure 16, highly complex geometries cannot be represented well by our limited resolution network, similarly the geometry cannot by a mesh decimated to equivalent memory footprint. These cases are simple to detect

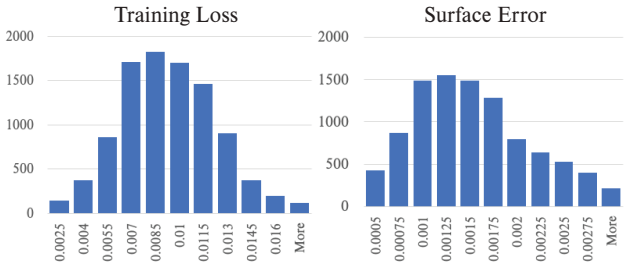

Figure 15: Loss and surface error distributions over the entirety of the Thingi10k [ZJ16] dataset.

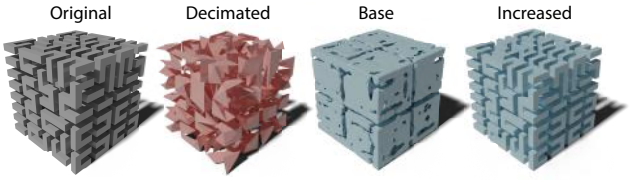

Figure 16: With only 7553 parameters, our *base Neural Implicit* format can lack the representative power to converge on highly complex geometries (similar to decimated mesh with same memory footprint). Increasing the network capacity to equal the memory impact of the original mesh results in near perfect reconstructions. tbuser (left) under CC BY.

by analyzing the surface error during training, and can easily be rectified by increasing the network resolution.

Neural networks have proven to be effective approximators of signed distance fields (SDFs) for solid 3D objects. Existing methods have seen success for generative modelling, shape interpolation, surface reconstruction, and differentiable rendering. However compelling, these methods share a common flaw, they depend on semantically consistent geometries in a fixed orientation. Instead of focusing training a model to generalize across some small subset of category, we overfit networks to single geometries applying the full network capacity to the reconstruction quality of a single shape. This not only allows for high quality reconstructions with minimal memory, but also supports any arbitrary geometry regardless of orientation or class.

We set out to show that overfitting a neural network to a single signed distance field is not only feasible, but can be useful in providing an efficient compact representation of the shape. We showed how our *Neural Implicit* representation can effectively capture a shapes signed distance function while maintaining a low memory impact and fast inference speeds. Our OVERFITSDF networks are a shape representation that inherit the effective infinite resolution of continuous implicits but with the computational efficiency of coarse meshes and the uniform memory patterns of an SDF grid. We hope that our geometrically principled approach to sampling training points and signing distances lead to more robust and extensible research in geometric deep learning.

References

[AL19] ATZMON, MATAN and LIPMAN, YARON. "Sal: Sign agnostic learning of shapes from raw data". *arXiv preprint arXiv:1911.10414* (2019) 2.

[BBB*97] BLOOMENTHAL, JULES, BAJAJ, CHANDRAJIT, BLINN, JIM, et al. *Introduction to implicit surfaces*. Morgan Kaufmann, 1997 1.

[BBL*17] BRONSTEIN, MICHAEL M., BRUNA ESTRACH, JOAN, LECUN, YANN, et al. "Geometric Deep Learning: Going beyond Euclidean data". English (US). *IEEE Signal Processing Magazine* 34.4 (July 2017), 18–42. ISSN: 1053-5888. DOI: 10.1109/MSP.2017.2693418 2.

[BDS*18] BARILL, GAVIN, DICKSON, NEIL, SCHMIDT, RYAN, et al. "Fast Winding Numbers for Soups and Clouds". *ACM Transactions on Graphics* (2018) 2, 5.

[CFG*15] CHANG, ANGEL X, FUNKHOUSER, THOMAS, GUIBAS, LEONIDAS, et al. "Shapenet: An information-rich 3d model repository". *arXiv preprint arXiv:1512.03012* (2015) 2.

[CZ19] CHEN, ZHIQIN and ZHANG, HAO. "Learning Implicit Fields for Generative Shape Modeling". *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2019) 4.

[GH97] GARLAND, MICHAEL and HECKBERT, PAUL S. "Surface simplification using quadric error metrics". *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co. 1997, 209–216 7.

[Har96] HART, JOHN C. "Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces". *The Visual Computer* 12.10 (1996), 527–545 5.

[HHF*19] HANOCKA, RANA, HERTZ, AMIR, FISH, NOA, et al. "MeshCNN: a network with an edge". *ACM Transactions on Graphics (TOG)* 38.4 (2019), 1–12 2.

[HSG18] HUANG, JINGWEI, SU, HAO, and GUIBAS, LEONIDAS J. "Robust Watertight Manifold Surface Generation Method for ShapeNet Models". *CoRR* abs/1802.01698 (2018) 2.

[HSW89] HORNIK, KURT, STINCHCOMBE, MAXWELL, and WHITE, HALBERT. "Multilayer feedforward networks are universal approximators". *Neural networks* 2.5 (1989), 359–366 3.

[JJHZ19] JIANG, YUE, JI, DANTONG, HAN, ZHIZHONG, and ZWICKER, MATTHIAS. *SDFDiff: Differentiable Rendering of Signed Distance Fields for 3D Shape Optimization*. 2019. arXiv: 1912.07109 [cs.CV] 2.

[JKS13] JACOBSON, ALEC, KAVAN, LADISLAV, and SORKINE-HORNUNG, OLGA. "Robust inside-outside segmentation using generalized winding numbers". *ACM Transactions on Graphics (TOG)* 32.4 (2013), 33 5.

[JPS*16] JACOBSON, ALEC, PANOZZO, DANIELE, SCHÜLLER, C, et al. *libigl: A simple C++ geometry processing library*. 2016 5.

[KB14] KINGMA, DIEDERIK P and BA, JIMMY. "Adam: A method for stochastic optimization". *arXiv preprint arXiv:1412.6980* (2014) 5.

[KMDT18] KERR, ANDREW, MERRILL, DUANE, DEMOUTH, JULIEN, and TRAN, JOHN. *CUTLASS: Fast Linear Algebra in CUDA C*. Sept. 2018. URL: https://devblogs.nvidia.com/cutlass-linear-algebra-cuda/ 6.

[LW19] LITTWIN, GIDI and WOLF, LIOR. "Deep Meta Functionals for Shape Representation". *CoRR* abs/1908.06277 (2019). arXiv: 1908.06277. URL: http://arxiv.org/abs/1908.06277 2, 3.

[LYF17] LIU, JERRY, YU, FISHER, and FUNKHOUSER, THOMAS A. "Interactive 3D Modeling with a Generative Adversarial Network". *Proc. 3DV*. 2017, 126–134 2.

[LZP*19] LIU, SHAOHUI, ZHANG, YINDA, PENG, SONGYOU, et al. *DIST: Rendering Deep Implicit Signed Distance Function with Differentiable Sphere Tracing*. 2019. arXiv: 1911.13225 [cs.CV] 2, 6.

[MAP*15] MARTIN ABADI, ASHISH AGARWAL, PAUL BARHAM, et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: http://tensorflow.org/ 5.

[MON*19] MESCHEDER, LARS, OECHSLE, MICHAEL, NIEMEYER, MICHAEL, et al. "Occupancy Networks: Learning 3D Reconstruction in Function Space". *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019 2, 5.

[MS15] MATURANA, DANIEL and SCHERER, SEBASTIAN. "VoxNet: A 3D Convolutional Neural Network for real-time object recognition". *Ieee/rsj International Conference on Intelligent Robots and Systems*. 2015, 922–928 2.

[NLBY18] NGUYEN-PHUOC, THU, LI, CHUAN, BALABAN, STEPHEN, and YANG, YONG-LIANG. "RenderNet: A deep convolutional network for differentiable rendering from 3D shapes". *Proc. NeurIPS*. 2018, 7902–7912 2.

[NMOG19] NIEMEYER, MICHAEL, MESCHEDER, LARS, OECHSLE, MICHAEL, and GEIGER, ANDREAS. *Differentiable Volumetric Rendering: Learning Implicit 3D Representations without 3D Supervision*. 2019. arXiv: 1912.07372 [cs.CV] 2.

[OBA*05] OHTAKE, YUTAKA, BELYAEV, ALEXANDER, ALEXA, MARC, et al. "Multi-level partition of unity implicits". *Acm Siggraph 2005 Courses*. 2005, 173–es 7.

[PFS*19] PARK, JEONG JOON, FLORENCE, PETER, STRAUB, JULIAN, et al. "DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation". *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019 1–6.

[QSMG16] QI, CHARLES R, SU, HAO, MO, KAICHUN, and GUIBAS, LEONIDAS J. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". *arXiv preprint arXiv:1612.00593* (2016) 2.

[WKBS19] WANG, YU, KIM, VLADIMIR G., BRONSTEIN, MICHAEL, and SOLOMON, JUSTIN. "Learning Geometric Operators on Meshes". *ICLR Workshop on Representation Learning on Graphs and Manifolds* (2019) 2.

[WSK*15] WU, ZHIRONG, SONG, SHURAN, KHOSLA, ADITYA, et al. "3D ShapeNets: A Deep Representation for Volumetric Shape Modeling". *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Boston, USA, June 2015 2.

[WSLT18] WANG, PENG-SHUAI, SUN, CHUN-YU, LIU, YANG, and TONG, XIN. "Adaptive O-CNN: A Patch-based Deep Representation of 3D Shapes". *ACM Transactions on Graphics (SIGGRAPH Asia)* 37.6 (2018) 2.

[WWX*17] WU, JIAJUN, WANG, YIFAN, XUE, TIANFAN, et al. "MarrNet: 3D Shape Reconstruction via 2.5D Sketches". *Advances In Neural Information Processing Systems*. 2017 2.

[ZJ16] ZHOU, QINGNAN and JACOBSON, ALEC. "Thingi10k: A dataset of 10,000 3d-printing models". *arXiv preprint arXiv:1605.04797* (2016) 3, 6–8.