Implicit High Resolution Representation of 3D Shapes

Fedor Sergeev[1*] Nicolas Talabot[2] Jonathan Donier[3] Pascal Fua[2]

[1]EPFL CSE [2]EPFL CVLab [3]Neural Concept

[1]fedor.sergeev@epfl.ch

June 10, 2022

Abstract

Deep implicit functions provide a powerful and flexible framework for encoding 3D shapes. Many previous works focused on training a deep neural network to represent multiple shapes, using an additional latent vector as input to differentiate between them. This approach produces representations of sufficient quality on average. However, in individual cases, the learned shapes often lack high resolution details. This limits the use of deep implicit functions for engineering and design, where precise shape encoding is crucial. In this work, we propose a simple setting for training a deep neural implicit to represent a single complex 3D shape and maintain its high-resolution features. On the particular shape we used, the reconstructed shape is visually more accurate then those obtained in previous works. Furthermore, we compare various approaches to designing the network architecture, sampling the implicit function values, picking the loss function, and identify the important parameters that affect the final quality.

1. Introduction

In computer vision, computer graphics, and engineering, explicit representations have long been used for encoding 3D shapes. While geometric representations, such as point clouds, voxel grids, and meshes, offer a robust and straightforward approach to storing shapes, they have a large memory footprint and present difficulties when used for continuous watertight shapes of complex topologies [10]. It is also challenging to use such approaches when the shape changes, for example when it is optimized with respect to some physical property (e.g., car surface optimized to reduce air drag). A reasonable transformation of a 3D shape could be used as an exploration technique in a first-order optimization method, for example, in kriging [8], or as a way to expand a dataset of labeled shapes used to train a GCNN [5].

[*]Semester project at EPFL CVLab.

A popular alternative approach to explicit geometric representation is the deep implicit representation. In this case, a deep neural network is trained to replicate an implicit function that represents the shape. A common choice of such function is the *Signed Distance Function (SDF)* [10]. For a continuous watertight shape, it is defined as

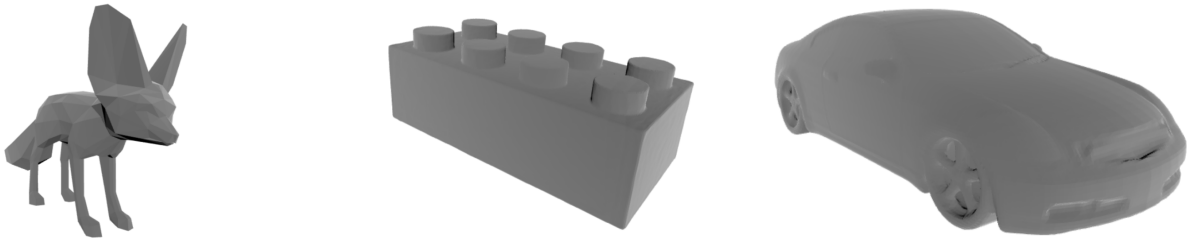
$$\text{SDF}(x) := s, \quad \boldsymbol{x} \in \mathbb{R}^3, s \in \mathbb{R}, \tag{1}$$

returning a distance s from a given spatial point $\boldsymbol{x}$ to the closest point of the surface, with a positive value outside the shape and a negative value inside. Thus, the underlying surface is represented by the zero isosurface $\text{SDF}(\boldsymbol{x}) = 0$, and can be rendered as a mesh using, for example, the Marching Cubes algorithm [9].

In previous works [3, 10, 13], the focus has largely been on creating models with a small memory footprint for representing a whole range of similar shapes with a good average precision. However, on close inspection, the individual shapes produced by such models often lack some physical features. For example, the spokes of a car are often not visible (see, for example, Figures 1 and 5 in [10]; Fig. 7a). In engineering applications, such details are crucial, as they might affect the physical qualities of a shape.

Our idea is to train a deep neural network network on a single shape, aiming for the best possible representation accuracy. This, in combination with the ability to warp the implicitly represented shape, as showcased in [13], would allow for optimization of 3D shapes while maintaining a high resolution. In this work we deal with the first stage of this process.

Thus, we build a deep neural implicit for representing a particular shape with high resolution. We study the effects of network architecture, SDF sampling strategy, and loss function choice on the quality of the learned representation. Using a combination of techniques from previous works, we obtain learned shapes that are almost visually identical to the original shapes.



(a) Fennec Fox [14] (b) Trilego [14] (c) Car [2]

Figure 1. Individual shapes used in this work.

2. Related works

The original paper that proposed using deep neural networks to learn an implicit representation is [10]. In addition to the coordinates of the SDF samples, the neural network also takes as input a latent vector, that encodes a particular shape from a dataset. This allows the model to represent multiple similar shapes without retraining (e.g., shapes from the same class in ShapeNet).

In a more recent work [13], it has been proposed to divide training into two stages: first, learning a general smooth template for the whole dataset (or object class) and second, warping the smooth template into a specific sharper final shape that is encoded with a latent vector. This paper gave us the original idea of splitting shape optimization into two stages: learning a high resolution template and warping the resulting SDF function into a new shape.

The weight-encoded neural implicit has been extensively studied in [3]. This work forgoes the use of latent vectors, instead training a model on a single shape, Furthermore, a novel method for sampling SDF values is presented. We test this sampling method and compare it with an alternative method in this paper.

There are alternative approaches to achieving higher resolution representation for complex shapes. One is to use high frequency signals as input: SIREN, positional encoding, and Fourier features [3]. Another is to train a model to represent local patches of the SDF, instead of the global shape [1]. We consider these techniques to be universal: they can be applied to our model as well, and most likely will lead to an increase in performance. Instead, we focus on a simpler setup to explore whether satisfying results are achievable.

In some cases, when direct visual comparison between shapes is not decisive, we use Chamfer [6] and Earth Mover's [12] distances (shortened to CD and EMD below) to compare the learned shapes with the original. The metrics measure the difference between the shapes and are defined as follows [10]:

$$\mathrm{CD}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} ||x - y||_2^2 + \sum_{y \in S_2} \min_{x \in S_1} ||x - y||_2^2,$$

$$\mathrm{EMD}(S_1, S_2) = \min_{\phi : S_1 \to S_2} \sum_{x \in S_1} ||x - \phi(x)||_2,$$

where S_1 and S_2 are point sets, ϕ is a bijection between two point sets of equal size. Following previous works [10], we normalize the Chamfer distance by the number of points in the point sets.

3. Methods

For the experiments, we used two test shapes from the Thingi10k dataset [14] and one car shape from the ShapeNet dataset [2] (see Fig. 1). All shapes are centered and normalized to fit into a unit sphere.

For particular settings, the training process works as follows. First, using a certain method, a set of points is sampled from the unit sphere around the shape. Then, at each epoch (or iteration), a random subset of these samples is used to compute the SDF values Eq. (1) and the loss function. The loss is subsequently differentiated, and the gradients are used to update the model weights.

Following [3, 10], we use deep fully-connected neural networks (FC NN) with ReLU activations in the intermediate layers and a $\tanh$ in the last layer, and the Adam optimizer with standard parameters [7]. We implement our models in Python using PyTorch [11]. The source code is available at github.com/TheodorSergeev/implicit-high-res-templates.

4. Experiments

There are 3 main components one has to consider when designing a deep neural implicit: the method for sampling SDF values, the architecture of the neural network, and the training procedure. For each of these components, we conduct experiments to identify the approach that yields the best quality.

First, we study the design of the network architecture. Second, we compare two different SDF sampling strategies

[3, 10]. Third, we investigate performance of different loss functions [4, 10].

4.1. Network architecture

4.1.1 Regularization

In previous work, network architectures were adapted to fit a whole class of shapes [10, 13]. To improve the generalizability of the models, regularization tools, such as dropout and normalization layers, were used. In our case, we effectively aim to purposefully overtrain the model on one shape. Thus the regularization features might actually reduce the quality of learned representation.

To investigate this hypothesis, we compare the performance of a fully connected neural network without regularization features and with the addition of dropout, skip connection, and weight normalization. In all cases the network, similarly to [3], consists with 8 layers of hidden size 32 followed by ReLU activations.

We used a car shape from the ShapeNet dataset (Fig. 1c), sampling the SDF values of 25000 points on the surface, 250000 points normally distributed around the surface with a variance of 10^{-3} and the same amount with a variance 10^{-4} (see Sec. 4.2 below for a detailed description of the sampling strategy). The model is trained with a simple L1 loss without clamping (see Sec. 4.3 below for a detailed description of the loss functions; no clamping is equivalent to $\delta = \infty$) for 10000 epochs using Adam optimizer with the initial learning rate of 10^{-3} and a ReduceLROnPlateau scheduler with a factor of 0.5 and patience of 200 (if the loss value does not decrease in 200 epochs by at least 10^{-4} of its value, the learning rate is reduced by a factor of 0.5).

As we see in Fig. 2, addition of weight normalization, and skip connection (to layer 4), as we expected, does not drastically improve the performance of the model. For dropout (0.2 in every intermediate layer) while the loss reaches a plateau, the model does not represent a correct SDF function, and so Marching Cubes does not recover a mesh.

To compare the quality of the models further, we compute Chamfer and Earth Mover's distances between the original and learned shapes (see Tab. 1). As before, there is no clear advantage in using or not using the regularization techniques.

Thus in the following experiments our architecture we decide to use the simplest architecture, consisting only of linear layers.

However, we should note that it is not completely ruled out that the skip connection will not show performance gains when used with deeper architectures, as shown in [10]. But, since in most of our experiments we use networks with 8 layers, we are not likely to see the performance benefits. Thus, for the sake simplicity, we decide not to use the skip connection.

Model	CD	EMD
Default	1.004	**0.087**
Weight normalization	79.63	0.296
Skip connection	**0.645**	0.098
Dropout	–	–

Table 1. Comparison of representation quality for models with different architecture using similarity metrics. CD = Chamfer Distance $\cdot 10^3$ (10000 points), EMD = Earth Mover's Distance (500 points).

4.1.2 Size

Next we investigate the effect of the network size on the quality of learned representation. As shown in [3, 10, 13], larger networks are able to achieve higher reconstruction accuracy.

Using the same setup as in the previous experiment, we study the effect of increasing the size of the neural network on the end quality. First, we increase the number of layers while keeping their width layer fixed. Second, we increase the width of layers while keeping their number fixed.

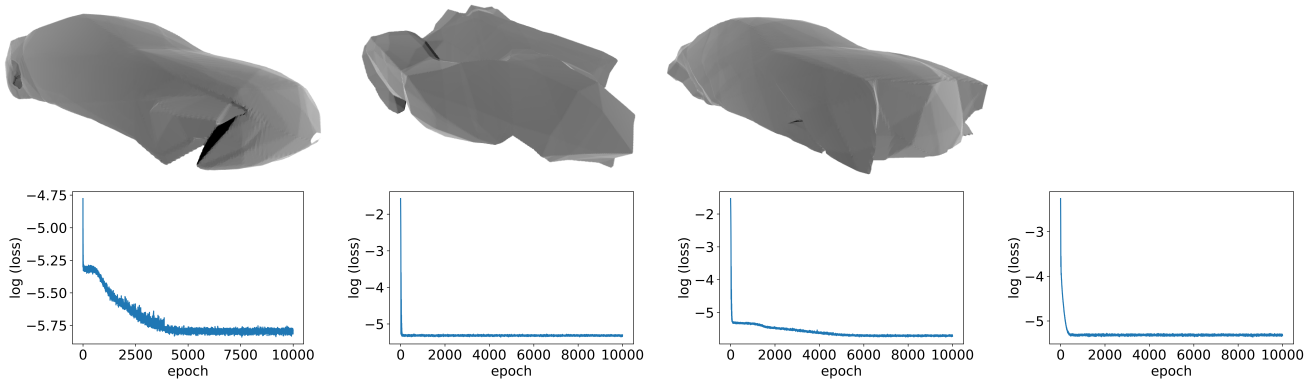
We find (see Fig. 3 and Fig. 4) that, as expected, increasing the size of the deep neural network leads to a higher accuracy of the learned representation. Notice, however, that each subsequent size of the model seems to yield smaller and smaller increases of quality, at least visually. For example, the change between a car shape for widths 512 and 2048 is barely noticeable. The possible reason is that the network already has the capacity to maintain most of the high resolution features of the shape, but is not trained quite to the point of full overtraining. Thus, in the following experiments we use the model with 8 layers and hidden size of 512, as it provides a good trade off between representation quality and speed.

4.2. SDF sampling

Sampling of the SDF values is a crucial part of the DeepSDF pipeline, as it is the direct protocol for creating the training set for a particular shape. Despite this, to the best of our knowledge, few attempts have been made to study the choice of different sampling scenarios and their effect on the quality of learned representations. The common intuition is that deep neural implicits benefit from aggressive sampling near the surface of the shape and a more sparse sampling further from the surface.

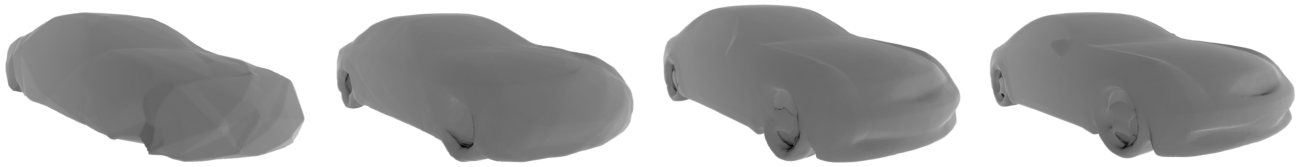
We compare two sampling strategies: *near surface sampling* and *weighted* sampling from [3].

Near surface sampling comprises a mixture of points from three distributions. The first distribution contains points that are directly on the surface of the shape. The second distribution consists of points from a centered normal distribution around random surface points. The third



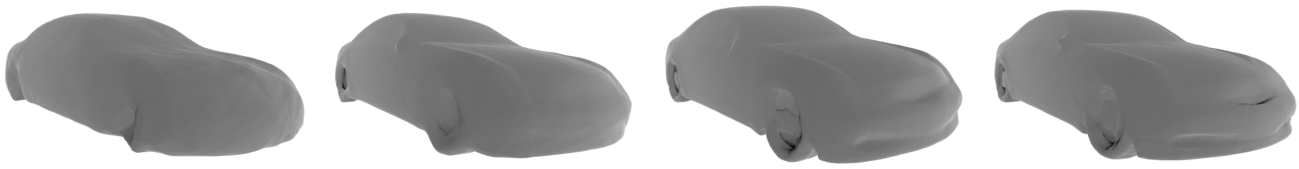
(a) Default architecture (b) With weight normalization (c) With skip connection (d) With dropout

Figure 2. Effect of regularization on quality of learned representation.



(a) 32 (b) 128 (c) 512 (d) 2048

Figure 3. Effect of increasing number of layers in a FC NN with a layer width of 512.



(a) 32 (b) 128 (c) 512 (d) 2048

Figure 4. Effect of increasing layer width for a FC NN with 8 layers.

distribution is equivalent to the second, except for a different variance of the normal distribution. Note that the first surface distribution can also be considered a normal distribution around the surface points with zero variance.

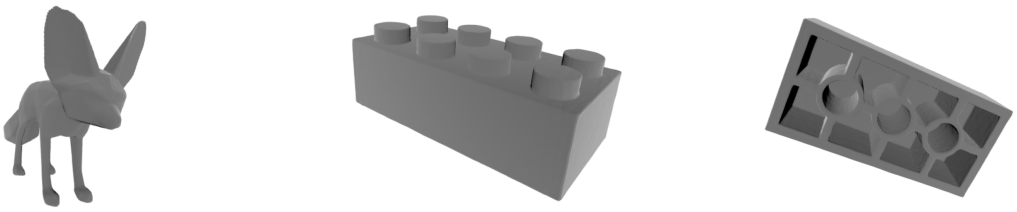
Weighted sampling is based on a simple subset rejection strategy. Starting with a large amount of uniform samples from the unit sphere around the shape, we resample a smaller subset according to a weighting function

$$w(\boldsymbol{x}) = \exp\left(-\beta \left|\text{SDF}(\boldsymbol{x})\right|\right), \tag{2}$$

where $\boldsymbol{x}$, such that $||\boldsymbol{x}|| \leq 1$, is the coordinate of a SDF sample. The parameter $\beta \geq 0$ can be adjusted to give more importance to samples close to the surface: when $\beta \to \infty$, only surface samples are selected, and when $\beta = 0$, the sampling is essentially uniform.

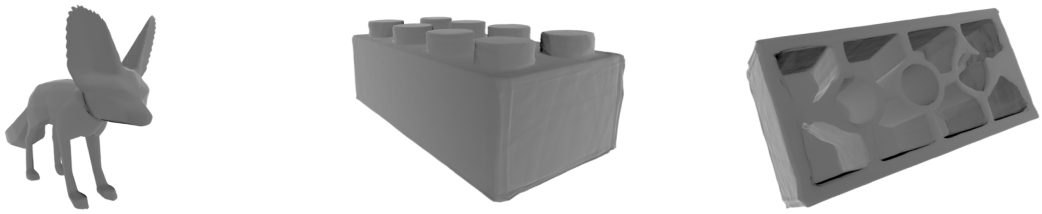
To compare the two approaches, we used the same setup as before to train a fully connected neural network on two shapes from the Thingi10k dataset. For the near surface sampler we use 25000 samples on the surface and 250000 samples for each of the normal distributions with variances 10^{-4} and 10^{-5}. For weighted sampling, we consider an initial pool of 10 million samples uniformly selected from a unit sphere, that is, subsequently to 1 million through weighted sampling with $\beta = 100$.

For the chosen parameters, the results are presented on figures 5 and 6. We see that for the Fennec Fox, both sampling techniques yield visually satisfying results. For the Trilego shape, while near surface sampling provides a clean and sharp shape, weighted sampling produces an askew shape, with one of the circles on the inner side of the lego block not connecting. We note that a saw-like pattern on the ears of the Fennec Fox is an artifact of visualization with Marching Cubes; it can be avoided with a higher resolution



(a) Fennec Fox (b) Trilego (c) Trilego

Figure 5. Shapes produced using the near surface sampling.



(a) Fennec Fox (b) Trilego (c) Trilego

Figure 6. Shapes produced using the weighted sampling.

grid.

Note that the choice of parameters of these sampling strategies is nontrivial. We select the parameters that seem to work best for these particular shapes and this particular network architecture. In our experiments we encountered convergence failure when using near surface sampling with a large number of surface samples and when using the weighted samples with large β's. To the best of our knowledge, choice of sampling parameters has not been systematically studied in previous works. For the near surface sampling we develop an intuition for selecting the number of parameters: for both normal distributions, choose the same number of samples, with second variance 10 times lower than the first one; for the surface distribution, number of samples should be around 5-15% of the number used for the normal distributions. For weighted sampling, we use the same number of samples as in [3], with a hand-picked β.

The convergence failure seems to occur when the deep neural implicit does not have enough samples that are relatively far from the surface. In this case, it is possible that the network does not find enough data to learn a valid SDF function. At the same time, the number of samples that are not close to the surface should be low, since capturing high resolution details largely depends on the availability of loss evaluations near those details.

We also notice that weighted sampling is much more computationally expensive compared to near surface sampling. Indeed, in the case 10^7 SDF values are computed and a subset rejection procedure is performed, where as in the second case only around $5 \cdot 10^5$ SDF values computations are required. Thus near surface sampling requires at least 50 times fewer SDF evaluations and avoids the subset rejection procedure overhead.

Informed by the obtained results and considerations of computational efficiency, we conclude that near surface sampling is both more effective and cheaper than the weighted sampling. We use near surface sampling in the following experiments.

4.3. Loss functions

When designing the training procedure, one needs to consider the choice of a loss function, the length of training, and the number of SDF samples used per training iteration.

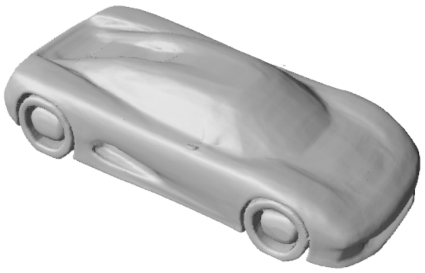
In our setting, the two latter aspects are rather easy to setup. Since we aim to purposefully overtrain the model, the training length should be as large as needed to reach a loss plateau. The number of SDF samples used per iteration is only limited by the amount of memory available on a particular GPU, and should in general be as large as possible to reduce variance of an optimization step. In our case, the training length is set to 10000 or 15000 epochs depending on the experiment, and the number of SDF samples per iteration is set to 10000.

As for the loss function, we consider two options: L1 loss with clamping from [10], and curriculum loss from [4].

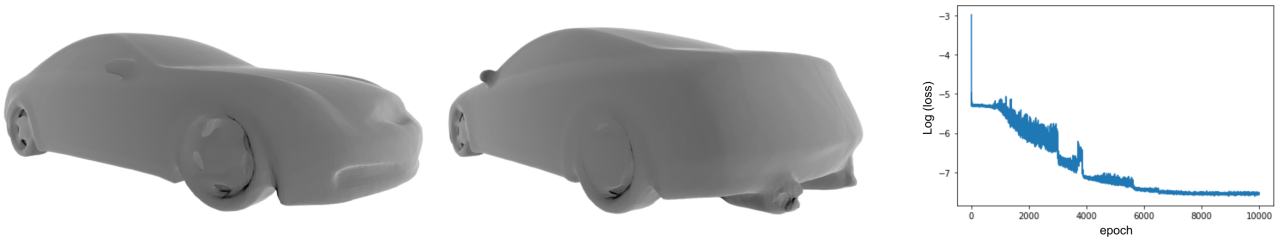
L1 loss with clamping is defined as

$$L(f_\theta(\boldsymbol{x}_j), s_j) = \left|\bar{s}_j - \bar{f}_\theta(\boldsymbol{x}_j)\right|, \tag{3}$$

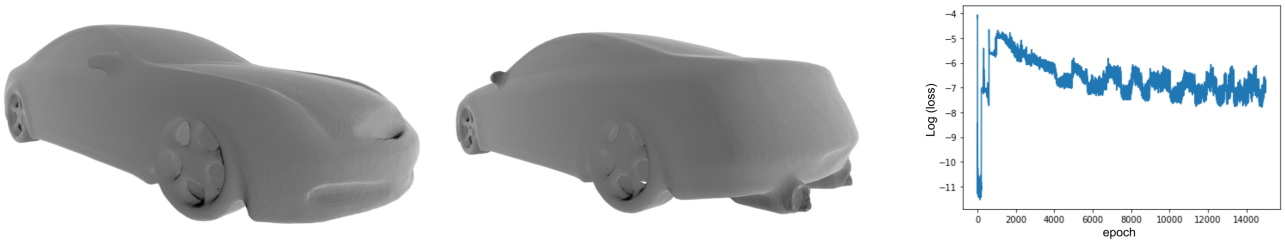
where $\boldsymbol{x}_j$ is the coordinate of a point in the unit sphere, s_j is the true SDF value for the point $\boldsymbol{x}_j$, f_θ is the model with



(a) Shape learned in the DeepSDF setting (Figure 5 in [10]).



(b) L1 loss with clamping and a scheduler



(c) Extended curriculum loss

Figure 7. Comparison of model performance trained with different loss functions.

parameters θ taking a coordinate $\boldsymbol{x}_j$ as input and outputting the predicted SDF value, and the bar $\bar{s}$ denotes the clamping function with a parameter δ. The clamping function is defined as

$$\bar{s} := \text{clamp}_\delta(s) = \min(\delta, \max(-\delta, s)).$$

Curriculum loss generalizes L1 loss with a smoothness coefficient ε and sample difficulty coefficient λ

$$L_{\varepsilon,\lambda}(f_\theta(\boldsymbol{x}_j), s_j) = k_\lambda(f_\theta(\boldsymbol{x}_j), s_j) \cdot L_\varepsilon(f_\theta(\boldsymbol{x}_j), s_j), \quad (4)$$

where

$$k_\lambda(f_\theta(\boldsymbol{x}_j), s_j) = 1 + \lambda \ \text{sign}(\bar{s}_j) \ \text{sign}\left(\bar{s}_j - \bar{f}_\theta(\boldsymbol{x}_j)\right),$$
$$L_\epsilon(f_\theta(\boldsymbol{x}_j), s_j) = \max\left\{L(f_\theta(\boldsymbol{x}_j), s_j) - \varepsilon, 0\right\}.$$

Setting $\varepsilon > 0$ at the start of training simplifies the optimization task, allowing the model to first learn a smoothed version of the shape. By reducing ε in the later stages of training, we force the model to reconstruct sharper versions of the shape. When the parameter is eventually set it to 0, the model should be able to represent the shape with high resolution.

Parameter $\lambda \in [0, 1)$ sets an importance level of samples for which the model made a mistake in the SDF sign. Such samples are usually located in the complex areas of the shape, and so an additional penalty on the loss from those areas makes the network capture finer details.

Start epoch	ε	λ
0	0.025	0.0
200	0.01	0.1
600	0.0025	0.2
1000	0	0.5
5000	0	0.8

Table 2. Parameters of the curriculum loss.

To compare the loss functions, we use the same network architecture and optimizer as before (FC network with 8 layers of hidden size 512, Adam with an initial learning rate of 10^{-3}) with the near surface sampling strategy (25000 samples on the surface, 250000 for each of the normal distributions with variances 10^{-4} and 10^{-5}). First, we train using the L1 loss with 0.1 clamping for 10000 epochs with the scheduler described in the previous sections. Second,

we train using an extended version of the curriculum loss for 15000 epochs without a scheduler (the parameters are listed in Tab. 2) [3, 4].

We see (figure 7) that both the L1 loss and the extended curriculum loss yield shapes that are sharper than those produced by the classic DeepSDF (notice, for example, separation of details in wheel rims) [10]. When comparing the L1 and curriculum shapes between each other, we clearly see that the curriculum approach produces better results. The greater detail is noticeable around the rims as well as sharper edges around the exhaust pipes. Since the difference is noticeable visually, we do not need to compare similarity metrics.

Note that while the L1 loss has reached a complete plateau in 10000 iterations, the curriculum loss has not. Since we do not use a scheduler for the curriculum loss, this provides an opportunity to extend the training even further, reaching event better results.

The intuition behind the curriculum loss achieving better results is that it reaches a comparable to L1 quality faster, and without the need to reduce the learning rate (instead we alter the curriculum parameters). This allows us to effectively train longer with the same learning rate, as well as have more meaningful gradients from complex areas of the shape thanks to the λ parameter.

However, we should note that constructing a curriculum is far from straight forward. For example, when changing the length of the training, one could expect to obtain better results by scaling the curriculum accordingly (e.g., first stage is 200 iterations for the overall training of 2000, so it would scale to 400 iterations for the overall length of 4000). In reality this produces much poorer results. Thus curriculum loss yields shapes that capture high resolution details, but it can be tricky to set up.

5. Conclusion

In this paper, we have proposed a setting for learning a high resolution neural implicit for a specific shape. We experimentally showed that a simple fully-connected neural network that is large enough can successfully capture fine details of 3D shapes when trained at length using a carefully constructed curriculum loss. We demonstrate that near surface sampling is a quick and intuitive SDF sampling approach that outperforms an alternative weighted sampling strategy. Learned shapes in our work are visually sharper and more detailed than some of those shown in previous works. These findings have the potential to provide a basis for a novel approach to efficient shape representation, transformation, and optimization.

References

[1] R. Chabra, J. E. Lenssen, E. Ilg, T. Schmidt, J. Straub, S. Lovegrove, and R. Newcombe. Deep local shapes: Learning local SDF priors for detailed 3d reconstruction. In *ECCV*, page 608–625, Berlin, Heidelberg, 2020. Springer-Verlag. 2

[2] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University – Princeton University – Toyota Technological Institute at Chicago, 2015. 2

[3] T. Davies, D. Nowrouzezahrai, and A. Jacobson. Overfit neural networks as a compact shape representation. *CoRR*, abs/2009.09808, 2020. 1, 2, 3, 5, 7

[4] Y. Duan, H. Zhu, H. Wang, L. Yi, R. Nevatia, and L. J. Guibas. Curriculum deepSDF. In *ECCV*, page 51–67, Berlin, Heidelberg, 2020. Springer-Verlag. 3, 5, 7

[5] N. Durasov, A. Lukoyanov, J. Donier, and P. Fua. DEBOSH: Deep bayesian shape optimization. *arXiv preprint arXiv:2109.13337*, 2021. 1

[6] H. Fan, H. Su, and L. Guibas. A point set generation network for 3d object reconstruction from a single image. In *CVPR*, pages 2463–2471, 07 2017. 2

[7] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 2

[8] J. Laurenceau and P. Sagaut. Building efficient response surfaces of aerodynamic functions with kriging and cokriging. *AIAA*, 46:498–507, 02 2008. 1

[9] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *SIGGRAPH*, 21(4):163–169, aug 1987. 1

[10] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *CVPR*, June 2019. 1, 2, 3, 5, 6, 7

[11] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *NIPS*, pages 8024–8035. Curran Associates, Inc., 2019. 2

[12] Y. Rubner, C. Tomasi, and L. Guibas. A metric for distributions with applications to image databases. In *CVPR*, pages 59–66, 1998. 2

[13] Z. Zheng, T. Yu, Q. Dai, and Y. Liu. Deep implicit templates for 3D shape representation. *CVPR*, pages 1429–1439, 2021. 1, 2, 3

[14] Q. Zhou and A. Jacobson. Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797*, 2016. 2