# Point2Weight

Júlia Garcia [1†] Joan Lafuente[1,2†] Maria Pilligua[1,2†]
[1] Universitat Autònoma de Barcelona [2] Computer Vision Center

## Abstract

*Point clouds are widely used for 3D object representation due to their simplicity and ability to capture real-world geometry. However, their unstructured, sparse, and often noisy nature makes them less suitable for many downstream applications. 3D surface reconstruction, transforming point clouds into continuous surface representations, is crucial for practical workflows in fields like computer graphics and engineering. Converting point clouds into implicit functions, such as signed distance functions (SDFs), offers advantages like high-resolution geometry and robustness to noise. While classical methods like Poisson Surface Reconstruction struggle with imperfect data, recent deep learning approaches like Points2Surf handle these challenges by learning robust priors. Building on this, we propose Point2Weight, a novel model using a HyperNetwork to predict the weights of an implicit function for a given input point cloud. Furthermore, we have conducted several ablation studies that build intuition about what the critical factors are to achieving superior reconstruction results.*

## 1. Motivation

Point clouds have become a widely used representation for 3D objects due to their simplicity, flexibility, and ability to directly capture the geometry of real-world scenes. They are commonly acquired from RGB-D sensors, structured-light scanners, LiDAR systems, or even from smartphones cameras, and serve as the raw output of many 3D scanning pipelines. However, point clouds are inherently unstructured, sparse, and often noisy, which makes them less suitable for many downstream applications that require surface-based or volumetric representations.

A key step in making point cloud data usable in practical workflows is 3D surface reconstruction—transforming the unstructured point cloud into a continuous representation of the object's surface. This process is crucial for a variety of fields, including computer graphics and content creation, archaeology, digital cultural heritage preservation, and engineering. In these applications, point cloud scans (which may come from casual mobile devices or high-precision laser scanners) must be converted into meshable representations for visualization, analysis, or fabrication.

One possible approach is to convert point clouds into implicit functions, such as signed distance functions (SDFs) or occupancy fields. These functions map 3D coordinates to scalar values indicating whether a point is inside or outside the surface. Implicit representations are particularly attractive because they naturally support high-resolution geometry, enable smooth surface reconstruction, and are amenable to gradient-based optimization and learning. They can be used to extract meshes via techniques like Marching Cubes, and are more robust to noise and missing data than traditional methods.

Several classical methods exist for this task, such as Poisson Surface Reconstruction [3], which attempts to fit an implicit surface to a given set of oriented points. While effective under ideal conditions, these techniques struggle in the presence of real-world imperfections like incomplete scans, sensor noise, or outliers. Common challenges when working with raw point clouds.

To address these limitations, recent methods like Points2Surf [2] have employed deep learning to learn robust priors over surface geometry. These models can better handle imperfect input data and generalize across a range of object categories. Following this direction, we propose Point2Weight, a novel model that makes use of a HyperNetwork to predict the weights of an implicit function for a given input point cloud.

Our contributions are as follows:

- We introduce Point2Weight, a novel approach that leverages a HyperNetwork to predict the weights of an MLP-based implicit decoder from point cloud embeddings.
- We enable per-object positional encoding by having the HyperNetwork also predict the parameters of a multi-resolution hash grid.
- We pretrain a PointNet using a classification pretext task to extract meaningful embeddings for conditioning the HyperNetwork.
- We conduct a comprehensive evaluation on the ABC dataset, demonstrating improved generalization and robustness to unseen and incomplete shapes.

[†]These authors contributed equally to this work.

## 2. State of the Art

Several strategies have been explored for surface reconstruction and implicit representation learning from 3D point clouds. In this section, we review classical geometry-based methods, modern deep learning approaches, and recent advancements in hypernetwork-driven models that relate closely to our proposed method.

**Poisson Surface Reconstruction (PSR)** [3] is a classical method that estimates a watertight surface by solving a spatial Poisson problem using the point normals. While efficient and robust to noise, standard PSR often struggles with thin structures and complex topology.

To improve its flexibility, techniques such as PSR with Envelope Constraints (PREC) [6] have been proposed. These constraints help guide the reconstruction to better preserve detailed features and avoid over-smoothing, especially in thin parts of the geometry.

**Learning-based methods** have emerged as powerful alternatives for surface reconstruction from raw point clouds. Points2Surf [2] introduced a deep implicit surface network that learns local shape priors and generalizes well to unseen geometries, allowing the model to output high-fidelity surfaces from raw point data.

More recently, **HyperNetwork-based methods** have been proposed to enable better generalization and adaptability across scenes or objects. HyP-NeRF [7] leverages hypernetworks to learn NeRF priors across multiple scenes, generating scene-specific NeRF weights conditioned on learned embeddings. This approach enables flexible reconstruction without retraining from scratch for each scene and highlights the potential of HyperNetworks to generalize implicit representations conditioned on input features.

Our proposed method builds upon these ideas by combining implicit function learning with a HyperNetwork that generates weights for both an MLP and a positional encoding module, tailored for each object. In contrast to HyP-NeRF which operates on radiance fields, our work focuses on learning Signed Distance Functions (SDFs) directly from point clouds.

## 3. Methodology

This section presents an overview of our method. We describe the evolution from a simple MLP-based architecture inspired by prior work to a hypernetwork based approach that predicts both the decoder weights and positional encodings. We also explain the PointNet pretraining strategy employed.

### 3.1. Learning Implicit Fields

Our initial approach was inspired by the architecture presented in *Learning Implicit Field for Generative Shape Modeling* [1], which proposes using implicit fields for learning generative models of shapes. This work introduced an implicit field decoder, known as IM-NET, designed for shape generation with the goal of enhancing the visual quality of generated shapes.

The core concept we adopted involved training a Multi-Layer Perceptron (MLP) with a given object embedding. This setup aimed to enable the model to learn multiple objects simultaneously and, ideally, to generalize to unseen objects. We conducted various experiments, modifying the MLP's layer configurations, changing its objective to predict either the signed distance function or the occupancy function, and altering the object embedding provided.

For embeddings, we experimented with both a pretrained PointNet and a learned embedding. Due to GPU constraints, we were unable to fine-tune PointNet during the main model's optimization process. Consequently, object features were precomputed, and the MLP was trained using these static features.

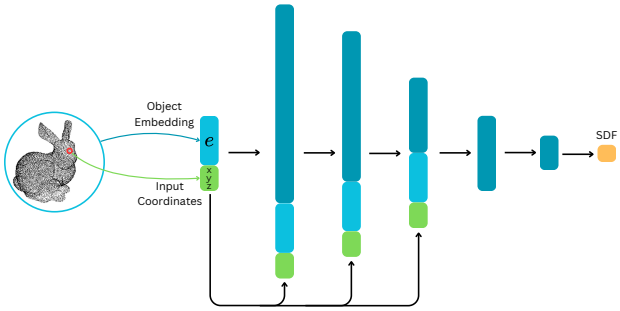


Figure 1. Arquitecture of our first attempt. Inspired from Chen et Al. [1], this showcases an MLP whose input is an object embedding $e$ and the input coordinates to be decoded $(x, y, z)$. The output is the signal distance function value of the input position.

### 3.2. HyperNetwork

Following our initial attempts, we completely shifted our approach by introducing hypernetworks. Hypernetworks are neural networks that predict the weights of another network, serving as a tool to add an extra layer of abstraction to the model. This is because instead of directly learning a fixed set of parameters that can work for any object, a hypernetwork learns to generate those parameters, effectively creating a specialized sub-network for each specific input or condition. So, in this case, rather than giving the object embedding to the main mlp like before, we now feed this embedding to a hypernet that predicts the weights of the mlp, and, therefore, this mlp is used as a container to put the weights and allow backpropagation to go though the pipeline, but what is actually being optimized is the hypernet.

The architecture can be seen in Figure 2. This second approach gives better results than the previous one. Some

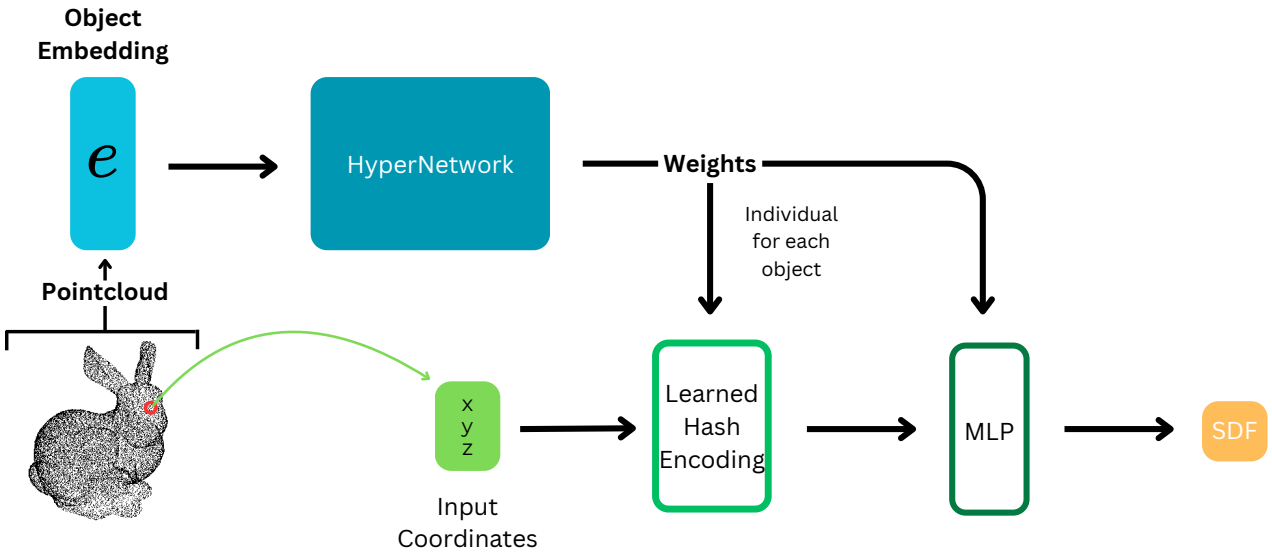


Figure 2. Architecture of the best performing model. Given an object to encode, we compute an embedding using PointNet and feed it to a hypernet. The hypernetwork then predicts the weights of a learned hash encoding and an MLP. This MLP corresponds to the main network that will be used to encode/decode the object. Given an input coordinate $(x, y, z)$ returns the signal distance function value that 3d object should have in order to represent the given pointcloud.

results can be seen in Section 4. The hypernetwork consists of an MLP of either 3 or 5 layers with hidden sizes of 256 or 512 respectively.

To counteract the common low-frequency bias observed in neural networks, we incorporated positional encoding. Our experiments indicated that a multi-resolution hash grid encoding, as described in [5], performed best. A crucial factor in improving our results was enabling the hypernetwork to also predict the weights for this multi-resolution hash grid encoding. This is significant because if the weights were not predicted by the hypernetwork, a single, common hash grid encoding would be learned for all objects, thereby limiting the model's flexibility. By contrast, when the hypernetwork predicts these weights, each new object benefits from its own uniquely learned positional encoding, allowing for a tailored fit in each case.

### 3.3. Embedding

We explored two primary approaches for generating object embeddings, visually detailed in Figure 3.

Object-Based Embedding: This method utilizes a standard learned embedding implemented in PyTorch. Each training example is assigned a unique ID, allowing the model to optimize its corresponding embedding to best represent the object for subsequent processing. This approach provides maximum flexibility for the model to learn object representations tailored to the task. While effective for assessing the architecture's capacity to memorize all training objects, a limitation of this method is its inability to decode objects not encountered during training.

**Point Cloud Based** Embedding: In this case, we use a PointNet pretrained using a pretext tasks (see Section 3.4). These embeddings are precomputed before training, and the PointNet is not optimized during training due to GPU limitations.

This approach allows to decode objects that were not seen during training and, ideally, would allow the model to generalize to new objects.


Figure 3. On the left, an object-based embedding, illustrating a PyTorch embedding where each training object is assigned a unique ID, and the network learns the optimal embedding for each. On the right, a Point Cloud-Based Embedding, where a pre-trained PointNet is utilized to embed each object.

### 3.4. Pretext Task

To extract robust and generalizable object embeddings from raw point clouds, we employed a pretext classification task using the PointNet architecture [8]. Specifically, the network was trained to classify 3D shapes from the ModelNet40 dataset [9].

**Dataset:** ModelNet40 contains 12,311 CAD models across 40 categories such as chairs, beds, monitors, etc. We followed the standard train/test split, and preprocessed the

dataset into fixed-size point clouds containing 2048 points per object, sampled uniformly from the surface mesh.

**Training Setup:** PointNet was trained with a standard negative log-likelihood loss ('NLLLoss') for classification, augmented with an orthogonality regularization term on the learned spatial transformation matrices to ensure geometric stability.

**Results:** The network achieved a classification accuracy of **85.14%** on the ModelNet40 test set. The confusion matrix (Figure 4) highlights minimal confusion between most classes. The main misclassifications occurred between semantically and geometrically similar categories such as *flower pot* vs. *plant*, and *curtain* vs. *door*.

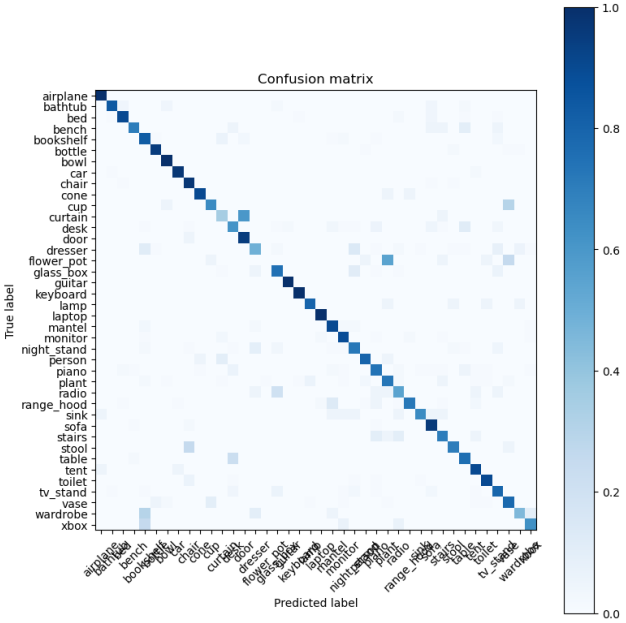


Figure 4. Confusion matrix on the ModelNet40 test set showing an overall accuracy of 85.14%. Misclassifications occur primarily between visually similar classes.

**Embeddings Visualization:** To assess the quality of the learned embeddings, we projected the latent vectors obtained into 2D using t-SNE. As shown in Figure 5, embeddings from the same class formed compact and separable clusters, confirming that the network learned semantically meaningful and discriminative representations, making it suitable for downstream tasks such as object-specific hypernetwork conditioning.

# 4. Experiments and Results

This section presents the experimental evaluation of our proposed method for implicit function estimation using the ABC dataset. We investigate various aspects of model performance, including memorization capacity, generalization

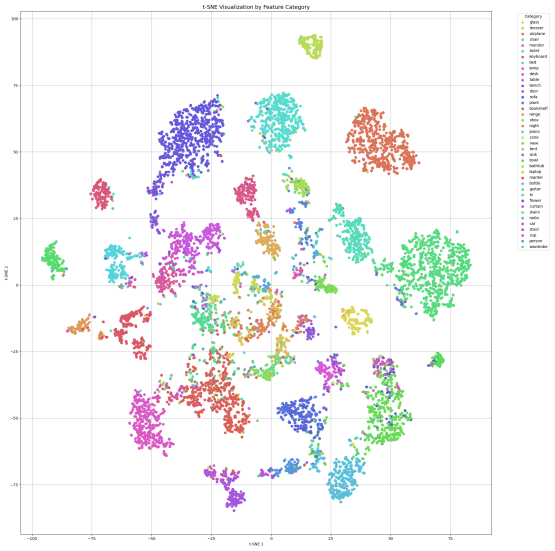


Figure 5. t-SNE projection of PointNet embeddings on ModelNet40 data. Each color corresponds to a different object category.

to unseen shapes, robustness to incomplete and noisy inputs, and computational cost.

## 4.1. Dataset

For the experiments of implicit function estimation, we have used subsets of the ABC dataset [4]. This dataset is a diverse collection of Computer-Aided Design (CAD) models for research on geometric deep learning methods and applications. In Figure 6, some examples of the models present in the ABC dataset can be seen, showing the diversity in the possible shapes.

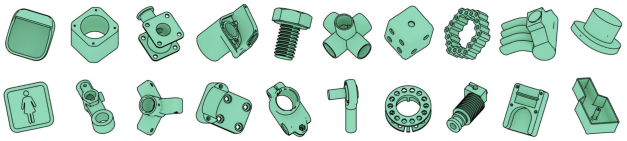

Figure 6. Some examples of ABC dataset models.

### 4.1.1. Sampling

To train a model to predict the implicit function—specifically, the signed distance function (SDF)—we need to provide it with a set of query points and their corresponding SDF values. These query points serve as training data, allowing the model to learn the shape represented by the SDF. A crucial step, then, is deciding how to sample these query points from a mesh.

We explored two different sampling strategies:

- Random Sampling: query points are sampled uniformly at random within a constrained region around the object.

- Proximity Sampling: 90% of points are sampled near the object's surface following a normal distribution centered at the boundary, with a standard deviation of 0.2. The remaining 10% of points are then sampled randomly from a constrained region surrounding the object.

We chose to use proximity sampling, as it concentrates more points near the object's surface. This provides the model with richer information about the object's geometry, which is especially valuable when learning the SDF. Figure 7 illustrates both sampling strategies used to extract point clouds from a mesh.

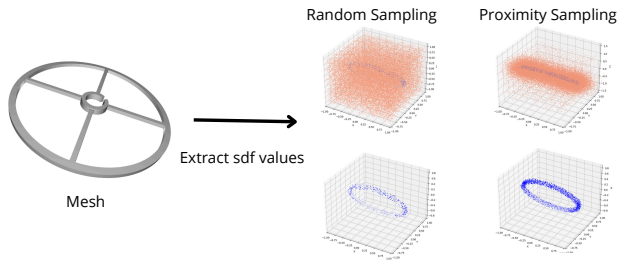


Figure 7. Visualization of the two sampling methods explored.

## 4.2. Memorization Capabilities

The first experiments done where focused in checking that the architecture made was capable of memorizing more than one object implicit function. This experiments started with a few objects, between 5 and 20, until the model could memorize all of them.

One key factor seen during these experiments was the need to predict the grid hash encoding using the HyperNetwork to be able to learn more detailed implicit functions, and be able to learn very thin objects. As explain in Sec. 3, this addition helped the model tackle the low frequency bias.

### 4.2.1. Training with 200 objects

Once we ensured that the architecture was capable of learning multiple objects we scaled up to a larger subset of ABC. This subset contained 200 random objects sampled from the ABC train set.

In this experiment we saw that the model could learn almost perfectly all of the objects seen during training, even those that had strange and thin shapes.

In Figure 8, we show some instances of the model's prediction and the corresponding ground truth. We only show the points with values $\leq 0$ for visualization purposes. Notice how the predicted query points within the object's boundaries (upper row) are close to the ground truth values (lower row).

### 4.2.2. Training with 5050 objects

At this point we checked the effects of increasing the amount of training data, as the target of this model is to be able to generalize to unseen objects. In these experiments, we saw that the model's performance on the training data was reduced. The model showed a bias towards low frequencies in some objects, having smoothed corners, and could not learn some small holes present in some objects.

In Figure 9, we can see some instances of this model prediction and ther corresponding ground truth.

## 4.3. Generalization Capabilities

An important characteristic that we aimed for in this project was that the model should be able to generalize to unseen objects' point clouds and estimate their implicit function correctly.

To evaluate this generalization capability, we tested the model on unseen objects and then visualized the predicted implicit functions. As shown in Figure 10 the model correctly estimates the orientation of the screw, but it has some minor errors across the object. At the same figure, we see something similar happening in a ring-like object, where the orientation is correctly estimated, but some errors occur near the object boundaries.

## 4.4. Robustness

One of the goals of this project is to develop a method that is more robust to noise and partial scans than traditional approaches, such as Poisson Surface Reconstruction. Although we were unable to train the model with specific data augmentations to enhance robustness, due to time constraints, we still wanted to assess whether the model's inherent characteristics could provide some initial resilience in these challenging scenarios.

### 4.4.1. Partial Scans

Traditional surface reconstruction methods struggle with partially scanned objects, as they lack the ability to infer whether missing regions are due to occlusions or actual object boundaries since they do not incorporate external knowledge or learned priors.

Our method could leverage information learned from the training data to understand the shape and know when there was a mistake when scanning the object. To test this idea, we simulated some partial scans by removing points that lay on one side of a manually set hyperplane.

During testing, we observed that in some cases—particularly with symmetric objects—the model was able to recognize partial scans and reconstruct shapes that closely resembled the original geometry. This is a highly desirable property for models of this kind, as real-world 3D scans often suffer from incompleteness due

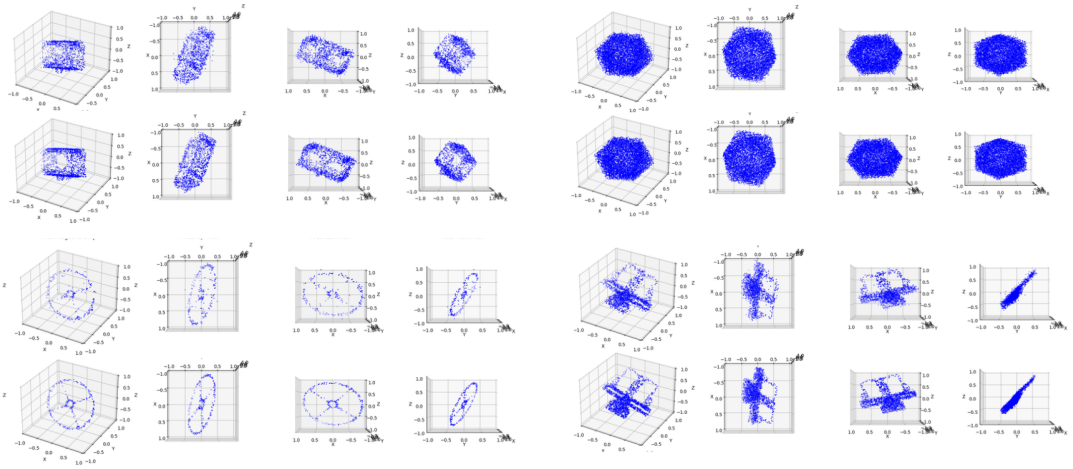

Figure 8. Four examples from the training set showing the ground truth and corresponding predictions of the model trained with only 200 objects. In each rectangle, the lower row displays the ground truth query points located within object boundaries, while the upper row shows the model's predictions.

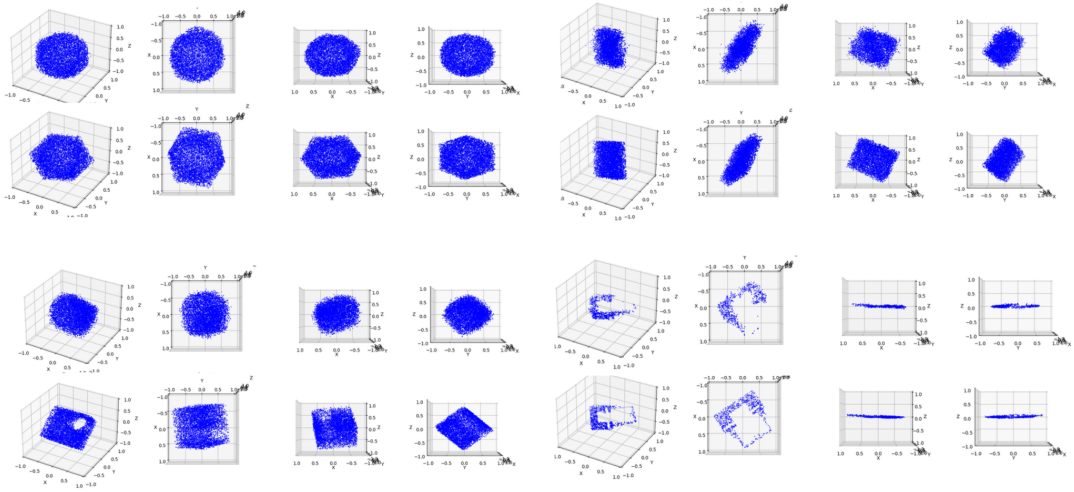

Figure 9. Four examples from the training set showing the ground truth and corresponding predictions of the model trained with 5050 objects. In each rectangle, the lower row displays the ground truth query points located within object boundaries, while the upper row shows the model's predictions.

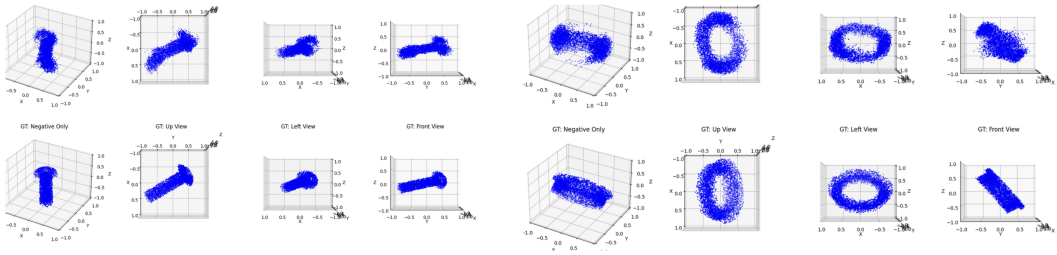


Figure 10. Example of the prediction of the model trained with 5050 objects, for two unseen validation objects point clouds. The lower row displays the ground truth query points located within object boundaries, while the upper row shows the model's predictions.

to occlusions or limited viewpoints during the scanning process.

This behaviour could be further improved if augmentations simulating these situations were performed during training.

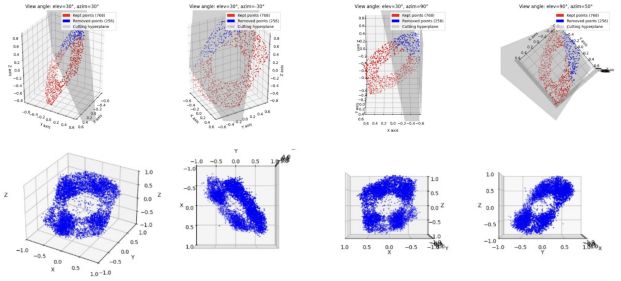

Figure 11. Visualization of a simulated partially scanned object and the corresponding model output (Model trained with 5050 objects).

#### 4.4.2. Noise

The appearance of noise in point clouds also poses problems for traditional methods. For a similar reason as before, these methods cannot understand when a point is noise or, is actually part of the object.

To simulate this situation in testing, we have added noise to some point clouds and visualized the model behaviour. In this case, we see a high degradation in the model outputs, as can be seen in Figure 12.

We believe these results are due to the fact that no component of the model—neither the PointNet encoder nor the hypernetwork—was exposed to noisy point clouds during training. As a result, the model lacks the ability to identify or correct for noise. Adding augmentations of noise during training would probably improve the robustness of the model to these perturbations.

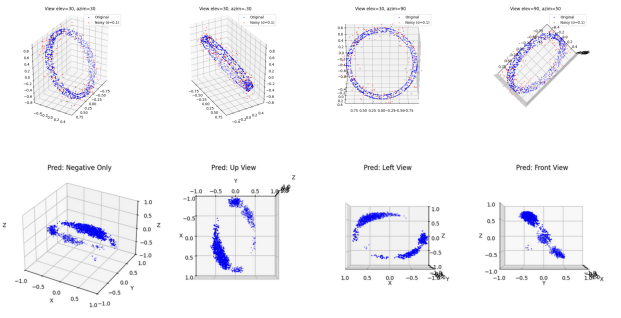


Figure 12. Visualization of a generated noisy point cloud of an object and the corresponding model output (Model trained with 5050 objects).

### 4.5. Computational Cost

So far, we have shown that predicting the hash grid encoding through a hypernetwork, providing object-specific encoding instead of learning a single global encoding, leads to better performance. However, this comes at the cost of an increased number of learned parameters and a higher computational load.

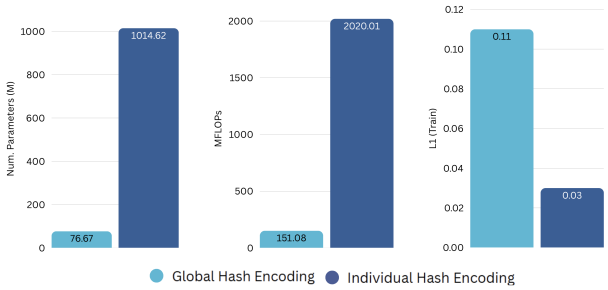


Figure 13. Comparison of computational cost and performance on the 200-object ABC subset between a global grid hash encoding and object-specific grid hash encodings generated via the hypernetwork.

As shown in Figure 13, using individual encodings for each object results in a model with approximately 10 times more learned parameters and operations. Nevertheless, this approach enables the model to achieve significantly better performance, as reflected in the lower L1 loss values.

### 4.6. Training Details

Our best-performing model consists of a hypernetwork composed of two MLPs: one dedicated to predicting the weights of the main network and another responsible for generating hash encodings. Both MLPs have four layers with a hidden size of 512. We use a multi-resolution hash grid encoding with 4 levels, a base resolution of 16, a target resolution of 2048, and a hashmap size of 15. This transforms the 3D input coordinates $(x, y, z)$ into a 256-dimensional embedding.

For training, we use a learning rate of $5 \times 10^{-4}$ for encoding, $1 \times 10^{-4}$ for the hypernetwork, and $1 \times 10^{-5}$ for the main network MLP. Optimization is performed using AdamW, with a step learning rate scheduler (step size of 10 epochs, decay factor $\gamma = 0.5$). We train with a batch size of 20,000 for 500 epochs using 5,020 objects from the ABC dataset. Training was completed in approximately 24 hours on an NVIDIA RTX 4090 GPU.

The final model comprises approximately 1.014 billion parameters and requires 2,020 MFLOPs for a single forward pass.

## 5. Conclusions

In this work, we presented a HyperNetwork-based approach for implicit function estimation from 3D point clouds. By conditioning both the weights of an MLP and a multi-resolution hash grid encoder on a PointNet-derived object embedding, our method effectively reconstructs accurate signed distance functions tailored to each object.

We demonstrated that the architecture could memorize and reconstruct a large number of training objects with diverse shapes. Moreover, our approach showed promising generalization to unseen validation objects, successfully estimating their implicit representations without retraining. Early experiments also revealed encouraging robustness against noise and partial scans.

Key factors contributing to our model's success include proximity sampling, which better captures object surfaces, and the generation of object-specific encodings via the HyperNetwork, helping to overcome the low-frequency bias common in neural networks.

As future work, we plan to explore: (i) combining multiple datasets to enhance shape diversity, (ii) incorporating noise and partial scans transformations to training data to improve robustness and generalization, and (iii) fine-tuning PointNet embeddings jointly with the HyperNetwork to better adapt to reconstruction tasks. Additionally, investigating alternative point cloud processing architectures like Point-Transformer could further boost representation quality and model flexibility.

## References


[1] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[2] Philipp Erler, Paul Guerrero, Niloy J Mitra, Tobias Ritschel, and Alexander Belyaev. Points2surf: Learning implicit surfaces from point clouds. In *European Conference on Computer Vision (ECCV)*, pages 108–124. Springer, 2020.

[3] Michael Kazhdan, M Bolitho, and Hugues Hoppe. Poisson surface reconstruction. *Proceedings of the fourth Eurographics symposium on Geometry processing*, 7(1):61–70, 2006.

[4] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[5] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, 2022.

[6] Kerem Yücel Oztireli and Markus Gross. Poisson surface reconstruction with envelope constraints. In *Computer Graphics Forum*, pages 633–642. Wiley Online Library, 2010.

[7] Keunhong Park, Matthew Tancik, Jonathan T Barron, and Ren Ng. Hyp-nerf: Learning improved nerf priors using a hypernetwork. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[8] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017.

[9] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1912–1920, 2015.