INRet: A General Framework for Accurate Retrieval of INRs for Shapes

Supplementary Material

7. Architecture and Training Details

7.1. INR Architecture and Training Detail

INR Training Losses. We apply different loss functions for different implicit functions. For the signed distance function, we follow the method in [42].

$$\mathcal{L}_s(f_\theta(\boldsymbol{x}), d_s(\boldsymbol{x})) = \|f_\theta(\boldsymbol{x}) - d_s(\boldsymbol{x})\|^2 \tag{5}$$

For the unsigned distance function, we follow the method in [11]

$$\mathcal{L}_u(f_\theta(\boldsymbol{x}), d_u(\boldsymbol{x})) = |f_\theta(\boldsymbol{x}) - d_u(\boldsymbol{x})| \tag{6}$$

For the occupancy field, we also apply an L1 loss similar to the unsigned distance function.

INR Architectures. We implement the INR architectures in this work using the NVIDIA Kaolin Wisp library [44]. We follow the default configurations for the NGLOD, iNGP and EG3D architectures. For NGLOD, we use an octree with 6 levels, but do not store features in the first 2 levels following the default configuration. We use a feature size of 8 for each level. For iNGP, we utilize 4 levels for the hash grid. The minimum and maximum grid resolutions are set to 16 and 512, respectively, with a maximum hashtable size of 2^{19} for each level. The feature size is 2 for each level. For EG3D, we use the default configuration with 1 level with a feature size of 8, but our solution could support multiple levels. For all grids, we initialize the grid features by sampling from a normal distribution with a mean of 0 and a standard deviation of 0.01. For the MLP-only INR, we follow the configuration in [13], and train a SIREN INR with 4 hidden layers and 512 hidden nodes [39]. The MLP uses sine activation functions.

INR Training. We use the polygon meshes from the ShapeNet10 and Pix3D datasets to generate SDF, UDF and Occ values to train the INRs. We use the sampling method from Kaolin Wisp. We sample 5×10^5 points for each shape per epoch of training. We sample 10^5 points uniformly in the domain $\Omega = \{\|\boldsymbol{x}\|_\infty \leq 1 | \boldsymbol{x} \in \mathbb{R}^3\}$, 2×10^5 on the surface of the shape, and 2×10^5 near the surface using normal distribution with a variance of 0.015. We use the same input coordinates for the training of all INRs. We train the grid-based INRs for 10 epochs and the MLP-only INRs for 100 epochs. We use Adam optimizer with a learning rate of $1e-3$ [21].

Compute Resources and Training Time All experiments and speed measurements were conducted on an Ubuntu 22.04 LTS system equipped with an Intel i7-13700K CPU and an NVIDIA RTX 4090 GPU. The retrieval and conversion times reported in Tab. 1. 2. 9. 19. were all measured on this system. We used Kaolin Wisp's implementation unless otherwise specified. We identified a significant inefficiency in Kaolin Wisp's point sampling algorithm, which performs point batching in plain Python. We addressed this issue by performing batching in PyTorch, resulting in approximately a 10x speedup. Consequently, training an INR and converting it to other data types takes about 1 minute in total. Evaluating our methods requires 12 INRs per shape (across 4 architectures and 3 implicit functions), leading to approximately 8.3 GPU days for INR training in the ShapeNet10 experiments alone. We anticipate that further improvements in INR modeling could reduce training times for even larger-scale experiments.

INR Initialization. For the MLP component of INR, we follow inr2vec's method and initialize the MLPs (of different INRs) with the same weights, as this has been proven essential for ensuring that the embedding from the MLP component is meaningful for shape retrieval [13]. However, we observe no such restriction for the initialization of the feature grids for different INRs, which is initialized with very small random values.

7.2. INR Encoder Details

INR MLP Encoder. We use an MLP encoder to convert the weights of the INR MLP into an embedding. For encoding MLP-only INRs, we use an encoder that is itself an MLP. This MLP encoder consists of 4 linear layers, each followed by batch normalization and a ReLU activation [19]. The final layer is a max pooling layer that produces an embedding of length 1024. For encoding the MLP component of a grid-based INR, we reduce the hidden size of all layers by half, using hidden layers with sizes 256, 256, 512, and 512. This results in an INR MLP embedding of length 512.

INR Conv3D Encoder. The Conv3D Encoder consists of five 3D convolution operations. Each convolution uses a kernel size of $(2, 2, 2)$ and a stride of $(2, 2, 2)$ to gradually reduce the spatial resolution. Each convolutional layer doubles the channel size and is followed by group normalization and a ReLU activation [50]. The final layer is a linear layer that maps the convolution output to an INR Grid Embedding of length 512. Combined with the INR MLP Embedding, the total INR Embedding length for the grid-based INR is 1024, which is the same as the embedding length for the MLP-only INR in inr2vec.

INR Encoder Training. We use the same input sampling process as in the INR training. Following the procedure in [13], we use the AdamW optimizer with a learning rate of 1×10^{-4} and a weight decay of 1×10^{-2} [24]. Note that

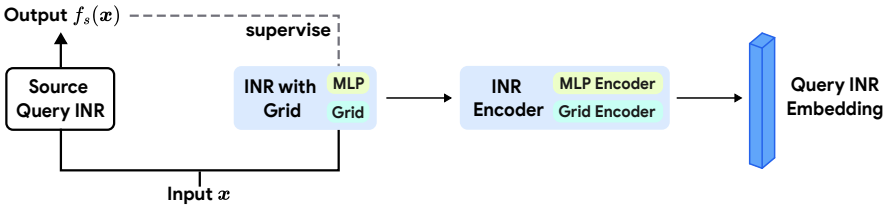
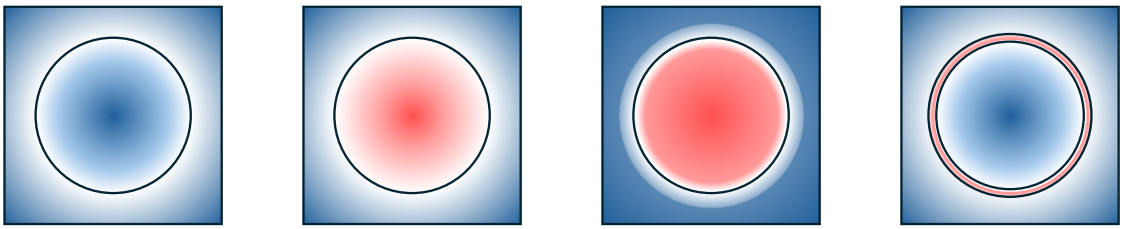


Figure 5. **INR Embedding Creation for INRs with Different Architectures**



(a) Single Layer Ball/Cylinder UDF **(b) Single Layer Ball SDF** **(c) Single Layer Ball Occ** **(d) Double Layer Cylinder UDF**

Figure 6. **Implicit Function Visual Representation for Cross Section of Different Shapes**

our decoder MLP f_ϕ has the same architecture as in [13]. Only the encoders are necessary for generating the INR embeddings. The decoder is used solely during the training of the encoders, and is not needed when encoding INR Embedding during inference.

7.3. INR Distillation

Changes such as modifications to the feature grid dimensions or the number of hidden nodes in the MLP can cause dimension mismatches, making encoders trained for specific architectures unusable. However, since one might need different architecture configurations for trade-offs between speed and representation quality, or use architectures that may be developed in the future, we need a general solution that does not have strict requirements on the INR architecture.

To address this, we leverage the property of INRs designed to output distance or occupancy values given an input coordinate. For a source INR with an unknown architecture, we create an embedding for retrieval by using the source INR f_s as an oracle. This involves generating pairs of input coordinates and output values from f_s to train an INR f_θ with an architecture compatible with our encoders. We refer to this as the INR distillation technique, as illustrated in Fig. 5.

$$f_\theta(\boldsymbol{x}; \boldsymbol{z}(\boldsymbol{x}, \mathcal{Z})) \approx f_s(\boldsymbol{x}),\ \forall \boldsymbol{x} \in \Omega. \tag{7}$$

In general, there are no limitations on the source or target INR architectures or the type of output value (distance or occupancy). In Sec. 5.3, we showed that distilling a source MLP-only INR to an INR with a feature grid can actually improve retrieval accuracy compared to using embeddings created from the MLP-only INR.

7.4. Explicit Representation Training and Encoding

PointNeXt. For training PointNeXt [35], we use point clouds containing 2048 points sampled from the surfaces of the INRs representing shapes in the training set. We follow the training procedure outlined in PointNeXt, using the PointNeXt-S variant. The training is supervised based on the shape class. After training, we remove the classification head and use the output from the PointNeXt backbone to create embeddings of length 512.

View-GCN. For training View-GCN [48], we use images rendered from the INRs representing the training shapes. We render 12 images at a resolution of 224 x 224 from virtual cameras positioned 3 units away from the object's center with a 0.65 elevation along a circular trajectory. We follow the training procedure specified in View-GCN, using shape class labels for supervision. After training, we remove the classification head and use the output of length 1536 as the embedding for shape retrieval.

7.5. Relationship Between Different Implicit Functions

At first glance, different implicit function fields may seem entirely distinct, even for similar shapes. For instance, the interior of a watertight shape is negative in an SDF repre-

Method	Ours		inr2vec	PointNeXt	View-GCN
Input Type	NGLOD	iNGP	MLP INR	Point Cloud	Multi-View Image
mAP @ 1	82.6	**84.2**	73.4	68.0	71.6
mAP @ 5	**94.4**	**94.4**	89.8	87.2	88.2
mAP @ 10	96.4	**96.6**	92.4	89.6	90.4
F1 @ 10	80.8	**81.8**	72.0	67.8	70.4

Table 5. Shape Retrieval Accuracy Metrics on ShapeNet10

Method	Ours		inr2vec	PointNeXt	View-GCN
Input Type	NGLOD	iNGP	MLP INR	Point Cloud	Multi-View Image
mAP @ 1	76.5	**78.0**	71.4	66.3	68.5
mAP @ 5	88.9	**93.8**	91.0	81.5	87.2
mAP @ 10	92.6	94.3	**95.5**	88.4	93.3
P @ 10	66.7	**68.0**	62.3	59.9	61.0
R @ 10	75.2	**76.1**	69.0	68.5	70.3
F1 @ 10	70.7	**71.9**	65.3	63.9	65.2

Table 6. Shape Retrieval Accuracy Metrics on Pix3d

		Retrieval INR		
		UDF	SDF	Occ
Query	UDF	69.4/70.4/66.7/61.2/68.5	71.6/72.8/12.2/59.9/66.4	71.6/71.6/10.7/60.8/61.2
	SDF	67.9/72.8/12.4/61.4/67.4	74.1/79.0/71.5/62.3/67.2	67.9/69.1/11.9/54.4/59.7
	Occ	69.1/67.9/13.1/57.6/60.4	72.3/74.1/11.8/56.8/60.9	68.9/69.1/65.4/58.2/61.3
Average		70.3/**71.9**/30.6/59.2/63.7		
Legend		NGLOD / **iNGP** / inr2vec / PointNeXt / View-GCN		

Table 7. Shape Retrieval Accuracy for Different Implicit Function INRs on Pix3D

		Retrieval INR		
		UDF	SDF	Occ
Query	UDF	80.2/83.0	91.2(+ 9.8)/88.4(+ 9.4)	86.6(+ 7.8)/87.4(+ 7.0)
	SDF	92.0(+ 9.8)/93.2(+11.4)	83.4/84.6	90.2(+11.0)/92.8(+10.4)
	Occ	85.6(+ 9.6)/89.4(+ 8.4)	87.8(+10.8)/92.6(+10.0)	76.8/83.0
Legend		NGLOD (+Improvement) / iNGP (+Improvement)		

Table 8. Shape Retrieval Accuracy for Different Implicit Function INRs on ShapeNet10, allowing Retrieval of Same Shape. In (bracket), we report the improvement in retrieval accuracy when retrieving the same shape is allowed.

sentation but positive in a UDF representation. This raises the question of how shape encoders can map INRs with different implicit function fields to a shared embedding space.

We analyze whether standard neural networks can relate different implicit function values. For the same underlying shape, the UDF, SDF, and Occ values are related by the following equations:

$$\text{UDF} = \text{ReLU}(\text{SDF}) + \text{ReLU}(-\text{SDF}) \tag{8}$$

$$\text{Occ} = \text{Sign}(\text{SDF}) \tag{9}$$

Eq. (8) utilizes standard summation, multiplication, and ReLU activations. Eq. (9) can also be calculated precisely using summation, multiplication, and ReLU activations, following these operations:

$$\begin{cases} h_1 = \text{ReLU}(\text{SDF}) \\ h_2 = \text{ReLU}(-\text{SDF}) \\ h_3 = \text{ReLU}(h_1 - 1) \\ h_4 = \text{ReLU}(h_2 - 1) \end{cases} \tag{10}$$

$$\text{Occ} = h_1 - h_2 - h_3 + h_4 \tag{11}$$

Our INR Encoders do not learn these mappings directly, as they operate with learned INR features at a global scale. However, Eq. (8) demonstrates that learning similar or even identical representations from UDF, SDF, and Occ is theoretically possible with standard neural network operations.

We visualize the differences between implicit functions of a simplified shape in Fig. 6. Consider the cross-section

	UDF	SDF	Occ
Point Cloud	1.26	0.98	1.82
Multi-View Images	4.13	3.05	3.67

Table 9. Conversion Speed (seconds) from INR with Different Implicit Functions to Different Representations

		Retrieval INR		
		UDF	SDF	Occ
Query	UDF	83.0/68.8/68.6	79.0/10.4/66.2	80.4/8.8/66.6
	SDF	81.8/11.4/67.8	84.6/70.2/70.0	82.4/10.4/67.4
	Occ	81.0/ 9.2/67.2	82.6/10.4/68.0	83.0/69.4/78.6
Average		82.0/29.9/67.8		
Legend		iNGP / MLP-only / MLP-only + INRet Regularization		

Table 10. Shape Retrieval Accuracy for Different Implicit Function INRs on ShapeNet10

of a watertight ball in Fig. 6(a) and (b). Fig. 6(a) and (b) show the SDF and UDF field respectively, and these fields can be simply related by Eq. (8). Fig. 6(c) is the typical learned Occ field of the same ball, where the values near the surface is zero, but +1 or -1 elsewhere. Note that the exact Occ field should be a solid fill both inside and outside the surface, but INRs often have trouble learning these exact values perfectly, and often learn near-zero values around a small region of the surface, which we show here.

Lastly, we show the SDF field of the cross-section of a double layer cylinder (open top and bottom) in Fig. 6(d). Compare this with Fig. 6(a), which is the UDF cross-section of a single layer cylinder, the UDF and SDF fields are almost the same everywhere except for the region in between the two layers. Note that SDF can be not be calculated for the single layer cylinder due to the lack of a watertight surface. This similarities shows that for the same or very similar shape, the underlying implicit function fields are also very similar, making learning the same embedding for the different fields easier.

8. Additional Results

In this section of the appendix, we provide additional results and ablation studies for INRet. App. 8.1 and 8.2 provides additional results for retrieval accuracy evaluation on ShapeNet10 and Pix3D. App. 8.3 demonstrates the effectiveness of INRet's regularizations on the retrieval of MLP-only INRs. App. 8.4 and 8.6 examines the impact of the implicit function of the *Unified Shape Decoder* on the final accuracy. App. 8.7 examines the impact of summation *vs.* concatenation of features from the spatial grids on the retrieval accuracy.

8.1. INR Retrieval with Feature Grids

In Tab. 5 and Tab. 6, we provide additional results and metrics for the experiment listed in Sec. 5.2. We show the mean Average Precision (mAP@k) at different numbers of k following the method in [13]. We also report the precision, recall, and F1 score following the definition in ShapeNet [37]. Note that for the ShapeNet10 dataset, since the number of models in each category is the same, the precision, recall and F1 score are the same, and thus we only report the F1 score. Our method achieves higher scores for almost all metrics across both ShapeNet10 and Pix3D datasets over inr2vec, PointNeXt and View-GCN. This demonstrates that our method is not only able to correctly retrieve the most similar shape, but also retrieves more shapes that belongs to the same category as the query shape as seen by the higher F1 score.

8.2. INRs with Different Implicit Functions

We show additional results for INR retrieval across different implicit functions on the Pix3D dataset in Tab. 7. Similar to the results on the ShapeNet10 dataset, our method demonstrates higher accuracy for retrieval across INRs with different implicit functions than inr2vec, PointNeXt and View-GCN.

Normally, we exclude the INR representing the same shape from being retrieved when measuring the mAP, otherwise, the query embedding always have the highest cosine similarity with itself. In Tab. 8, we show the accuracy of INR retrieval across different implicit functions by allowing retrieval of INR (with a different implicit function) representing the same shape. As seen in the table, there is around 10% improvement in retrieval accuracy. This shows that in many cases, the retrieved shape is the same shape as the query shape, but just represented with a different implicit function.

In Tab. 9, we show the conversion speed of converting different representations to point clouds and multi-view images. As required by View-GCN, 12 images are rendered, and as a result, it is more expensive than sampling a single point cloud. Conversion to point cloud or images is also more expensive for UDF compared to SDF due to the use of damped spherical tracing.

		Retrieval INR		
		UDF	SDF	Occ
Query	UDF	**83.0**/80.8/79.4	79.0/81.2/78.8	80.4/79.8/80.4
	SDF	81.8/81.2/80.0	84.6/**85.8**/83.6	82.4/82.4/82.6
	Occ	81.0/79.6/80.8	82.6/**82.8**/81.4	83.0/83.2/**83.4**
Average		**82.0**/81.9/81.2		

Table 11. Shape Retrieval Accuracy for iNGP INRs on ShapeNet10 with Different *Unified Shape Decoder* Implicit Functions (UDF/SDF/Occ)

		Retrieval INR		
		UDF	SDF	Occ
Query	UDF	83.0/82.6/82.0/82.8	79.0/80.2/78.4/80.4	80.4/80.6/81.0/79.8
	SDF	81.8/81.8/82.0/81.0	84.6/84.0/83.8/84.8	82.4/81.4/81.8/80.8
	Occ	81.0/81.2/80.8/80.6	82.6/82.0/83.0/82.6	83.0/82.6/83.2/82.8
Average		**82.0**/81.8/81.8/81.7		

Table 12. Shape Retrieval Accuracy for iNGP INRs on ShapeNet10 with Different Explicit L2 Regularization Weights for *UDF-SDF, UDF-Occ, SDF-Occ* (*111/211/121/112*)

		Retrieval INR		
		UDF	SDF	Occ
Query	UDF	83.0/82.5	79.0/81.0	80.4/80.6
	SDF	81.8/81.6	84.6/84.6	82.4/81.8
	Occ	81.0/81.4	82.6/79.0	83.0/79.6
Average		**82.0**/81.3		

Table 13. Shape Retrieval Accuracy for iNGP INRs on ShapeNet10 with UDF *Unified Common Decoder* L2/L1 Loss Choice

8.3. Applying INRet Regularization to MLP-only INRs

In Tab. 10, we demonstrate the retrieval accuracy when we apply INRet unified latent space regularizations (L2 + *Unified Shape Decoder*) to MLP-only INRs. We also include the iNGP retrieval accuracy for comparison. As seen from the table, the retrieval accuracy of MLP-only INRs significantly increases when the unified latent space regularizations are applied. This shows that our regularization techniques apply to both INR with and without feature grids. However, for the MLP-only INRs, the final accuracy is still lower than if the iNGP INR with feature grid is used to create the INR embeddings.

8.4. Choice of Unified Shape Decoder Implicit Function

In this section, we evaluate the performance of INRet with different Unified Shape Decoder implicit functions. In Sec. 5.4, we used the UDF as the implicit function for the *Unified Shape Decoder*. In Tab. 11, we show the retrieval accuracy when the unified decoder outputs different alternative implicit functions during training.

As seen in Tab. 11, the average retrieval accuracy for different choices of common decoders is fairly close. The UDF common decoder had the highest accuracy of 82.0% while the lowest, the Occ common decoder, is only 0.8% behind in accuracy. However, we observe that if the INR's implicit function is the same as the common decoder's output, the retrieval accuracy tends to be higher. For example, for SDF to SDF retrieval, the highest retrieval accuracy of 85.8% is achieved when the common decoder's output is also an SDF. The trend also applies to UDF to UDF retrieval and Occ to Occ retrieval. In addition, for retrieval across INRs with different implicit functions, the retrieval accuracy tends to be higher if the query or retrieval INR's implicit function is the same as the common decoder's output type. Despite these tendencies, our method is generally relatively robust to the choice of the common decoder's output.

8.5. L2 Regularization Weight

In this section, we evaluate the performance when different weights are applied to the explicit L2 regularization. In INRet, the explicit L2 regularization is simultaneously applied to 3 different pairs: UDF-SDF, UDF-Occ and SDF-Occ. In the main results presented in the paper, the weighting is the same for all pairs. In Tab. 12, we show the retrieval accuracy when the weights are different. For example, the *211* weight means the UDF-SDF loss is multiplied by 2 before being added to the total loss, while the UDF-Occ and SDF-Occ are multiplied by 1.

From Tab. 12, we observe that our method is robust with

respect to the specific choice of weight multipliers. For the INR encoder training, we used the Adam optimizer which has an adaptive learning rate on individual weights of the network, eliminating the need for careful fine-tuning on the weight multipliers [21].

8.6. Norm Choice for Unified Shape Decoder

For a specific implicit function, our choice of norm simply follows that used in existing works. As explained in Appendix 7.1, we follow the method in [11] and use the L1 normalization for both UDF INR training and when the *Unified Shape Decoder's* implicit function is UDF. In this section, we test whether using L2 normalization for the *Unified Shape Decoder* (with UDF implicit function) instead of L1 has an impact on the accuracy. We present the results in Tab. 13. From the table, we show that using L2 normalization decreases the retrieval accuracy slightly compared to using the L1 loss on average. We note that the retrieval accuracy of individual loss function to loss function pairs can fluctuate quite significantly. For example, the Occ-SDF retrieval accuracy dropped 3.6% (from 82.6% to 79.0%). This is different from the result in Tab. 12 where the weighting of the explicit regularization had minimal impact on the retrieval accuracy.

8.7. Summation and Concatenation of Features

In the main results, the Conv3D encoder takes in the summation of features from NGLOD feature grid and the concatenation of features from iNGP feature grid. We do so because summation and concatenation of features are used in the original NGLOD INR and iNGP INR respectively.

Following Eq. (2), for NGLOD, the features from the multi-level octree feature grid are summed before fed into the MLP, i.e.

$$\boldsymbol{z}(\boldsymbol{x}, \mathcal{Z}) = \sum_{l}^{L} (\boldsymbol{\psi}(\boldsymbol{x}; l, \mathcal{Z})) \tag{12}$$

For iNGP, the features are concatenated instead, i.e.

$$\boldsymbol{z}(\boldsymbol{x}, \mathcal{Z}) = [\boldsymbol{\psi}(\boldsymbol{x}; 0, \mathcal{Z}), \boldsymbol{\psi}(\boldsymbol{x}; 1, \mathcal{Z}), \ldots, \boldsymbol{\psi}(\boldsymbol{x}; L, \mathcal{Z})] \tag{13}$$

For NGLOD, features stored in different levels of the octree capture varying levels of geometry detail. Therefore, using summation allows adding finer surface information (deeper level) to the coarser overall shape (upper level) [42]. For iNGP, the features stored in the hash grid inevitably suffer from hash collision. The authors argued that using features from all levels would allow the MLP to mitigate the effect of hash collision dynamically [29]. Using the octree feature grid, NGLOD does not suffer from the hash collision issue.

Following the experiment setting listed in Sec. 5.2, we test the retrieval accuracy when we use features in a way different from how it was used in the original INR architecture.

As shown in Tab. 14, both methods experienced a significant drop in retrieval accuracy if the features were not used in a way consistent with the original INR. For NGLOD, the concatenation of features leads to an accuracy drop of 14.8%. We note that the concatenation of features in this case actually means more features being passed to the Conv3D encoder for INR embedding creation. However, since the finer level features were never used alone in the original NGLOD INR training, we hypothesize the Conv3D encoder may be overfitting to these finer level features that might be noisy when used standalone. For iNGP, the retrieval accuracy is dropped by 53.8% since the summation of features leads to a significant loss of information.

8.8. Additional Retrieval Visualization

We show retrievals that failed to retrieve from the same category in Fig. 7. As seen in the figure, given a query chair, the retrieved examples can be from other categories albeit resembling some semantic similarities with the query itself.

Figure 7. Chair Retrieval Incorrect Classes

9. the Impact of Reconstruction Quality on Retrieval Accuracy

9.1. Reconstruction Quality

In this section, we provide additional details on the quality of reconstruction of the trained INRs with respect to the original mesh. For UDF INRs, we measure the Chamfer Distance (C.D.) at 130,172 points, following the same sampling method used in [42]. However, instead of regular spherical tracing, we apply damped spherical tracing similar to [11]. For SDF and Occ INRs, we use vanilla spherical tracing without damping, and we also measure generalized Intersection over Union (gIoU) which calculates the intersection of the inside of two watertight surfaces with respect to their union. We do not measure gIoU for UDF INRs as there is no notion of inside and outside.

As seen in Tab. 15, both the NGLOD and iNGP achieve higher reconstruction quality than the MLP INRs. These INRs with feature grids are not only superior at representing shapes with higher fidelity but also lead to higher retrieval accuracy.

INR Arch.	NGLOD		iNGP	
Feature Comb.	Sum (Original)	Concat (Modified)	Concat (Original)	Sum (Modified)
mAP @ 1	**82.6**	67.8	**84.2**	30.4

Table 14. Shape Retrieval Accuracy on ShapeNet10 when Features are Summed or Concatenated from the Feature Grids

INR Arch.	NGLOD			iNGP			MLP		
Implicit Func.	SDF	UDF	Occ	SDF	UDF	Occ	SDF	UDF	Occ
C.D. ShapeNet	0.0168	0.0122	0.0210	0.0147	**0.0119**	0.0223	0.0354	0.0344	0.0389
C.D. Pix3D	0.0183	0.0125	0.0213	0.0146	**0.0120**	0.0241	0.0367	0.0351	0.0392
gIoU ShapeNet	84.2	NA	81.4	**86.2**	NA	82.1	77.3	NA	75.2
gIoU Pix3D	85.5	NA	82.2	**86.5**	NA	82.3	77.5	NA	74.9

Table 15. Shape Reconstruction Quality of different INRs on ShapeNet and Pix3D

9.2. Reconstruction Quality and Retrieval Accuracy

Since INR with feature grids have both higher reconstruction quality and higher retrieval accuracy, one may wonder if these are correlated. We perform another experiment, where the iNGP is only trained for only 2 epochs, leading to reconstruction quality lower than the MLP-only INR. As seen in Tab. 16, the retrieval accuracy for iNGP significantly drops when the INRs are undertrained. However, retrieval with iNGP @ 2 epochs still has 5.4% higher accuracy compared to retrieval with MLP-only INR. The MLP-only INR lacks the features stored spatially in the feature grid, which is very useful for improving retrieval accuracy.

10. Retrieval of INRs trained using Different Source Data

In Section 5.4, we demonstrated the retrieval accuracy across different INR implicit functions. These implicit functions are trained using the same source information (meshes). In Tab. 17, we show another case where the UDF INRs are trained using point clouds sampled from the meshes instead of using the meshes directly [11]. As seen in Tab. 17, the retrieval accuracy is very similar regardless of the type of the source training data, showing that INRet can enable the retrieval of INRs when the INRs are trained with different inputs.

11. Category-Chamfer Metric

11.1. Retrieval Accuracy by Category and Chamfer Distance

Shape retrieval performance is traditionally evaluated based on whether the retrieved shape has the same category as the query shape [37]. While this metric can evaluate the quality of retrieval based on overall shape semantics, it largely ignores similarities or differences between individual shape instances. To mitigate the shortcomings of existing metrics, we propose the Category-Chamfer metric, which evaluates whether the retrieved shape shares the same category as the query shape, and also has the least Chamfer Distance with respect to the query shape for all shapes in the category.

We choose Chamfer Distance as it can measure the distance between almost all 3D representations. Chamfer Distance is a metric that calculates similarity between two point clouds. Unlike certain metrics such as generalized Intersection over Union (gIoU) that require watertight surfaces with a clear definition of inside and outside, Chamfer Distance only requires a point cloud which can be easily converted from other 3D representations including meshes, voxels, and INRs.

The accuracy A_C based on category information only is

$$A_C = \frac{\sum_{q \in Q} \delta(C(q), C(R(q)))}{|Q|} \tag{14}$$

where Q is the query set, C and R denote the category and retrieval function respectively, the Kronecker delta $\delta(\cdot, \cdot)$ evaluates to 1 if $C(q)$ and $C(R(q))$ are the same and 0 otherwise. The accuracy is normalized by the total length $|Q|$ of the query set.

The accuracy A_{CC} based on category and Chamfer Distance is

$$A_{CC} = \frac{\sum_{q \in Q} \left[\delta(C(q), C(R(q))) \times \delta\left(s', C(R(q))\right)\right]}{|Q|}$$
$$\text{where } s' = \underset{s \in S}{\operatorname{argmin}}\, d_{CD}(q, s) \tag{15}$$

where d_{CD} denotes the Chamfer Distance, S denotes the set of all candidates for retrieval.

Category-Chamfer is a more challenging metric compared to category-only metric, in our experiments, we find that we can leverage the Chamfer Distance between the the INR instances to achieve a high accuracy for this metric.

11.2. Category-Chamfer Retrieval Accuracy by Embedding Cosine Similarity

Compared with the category-only accuracy, achieving high accuracy as measured by the Category-Chamfer metric is

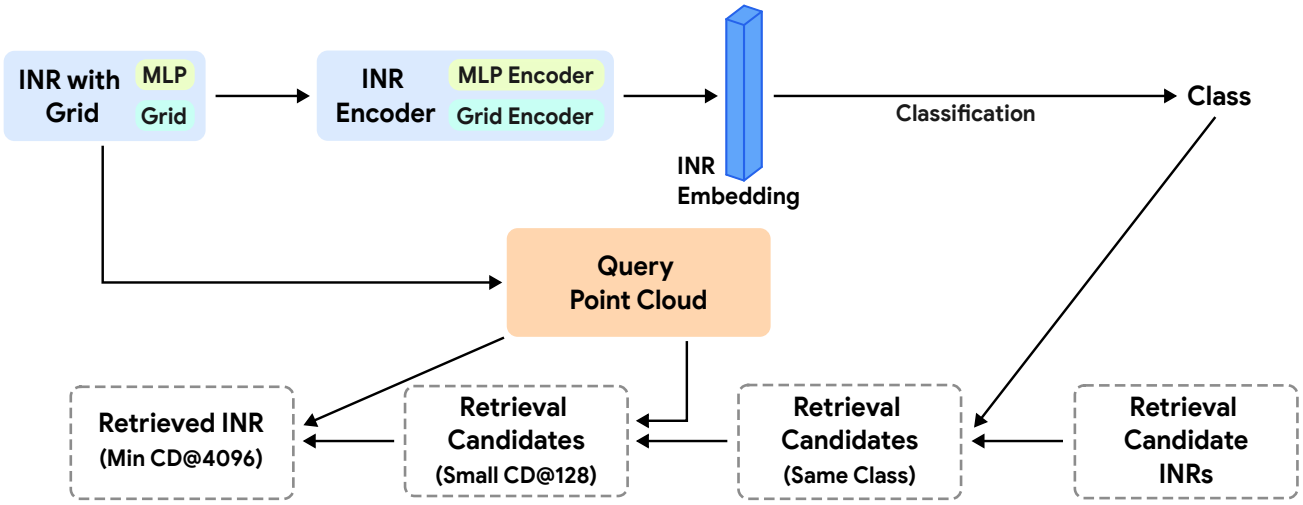


Figure 8. Hierarchical Sampling Retrieval Method

Method	Ours		inr2vec
Input Type	iNGP	iNGP @ 2 Epoch	MLP INR
mAP @ 1	**84.2**	<u>78.8</u>	73.4
C.D.	0.0168	0.0371	0.0354

Table 16. Shape Retrieval Accuracy and Reconstruction Quality Comparison for Different INR Architectures (SDF) on ShapeNet10

		Retrieval INR		
		UDF	SDF	Occ
Query	UDF	80.2(−0.2) / 83.0(+0.0)	91.2(+0.2) / 88.4(−0.2)	86.6(+0.0) / 87.4(−0.4)
	SDF	92.0(+0.0) / 93.2(−0.2)	83.4 / 84.6	90.2 / 92.8
	Occ	85.6(+0.0) / 89.4(+0.0)	87.8 / 92.6	76.8 / 83.0

Table 17. Shape Retrieval Accuracy for Different Implicit Function INRs on ShapeNet10 (Accuracy Change when UDF INRs are trained using point clouds instead of meshes)

more challenging. By simply comparing the cosine similarity between embeddings, neither INRet or existing methods such as PointNeXt perform well for this new metric. We exclude View-GCN from this evaluation since it may not require an actual 3D model to perform the retrieval and thus may not be able to calculate Chamfer Distance given its input. Following the procedure in Sec. 5.3, we evaluate the Category-Chamfer accuracy.

We calculate the ground truth Chamfer Distance at 131072 points following the same sampling method from [42]. From Tab. 18, we observe that the Category-Chamfer accuracy for all methods is very low. The highest accuracy is achieved by PointNext at 28.4%, far below its category-only accuracy of 71.2%. In the next section, we provide a solution for increasing the Category-Chamfer retrieval accuracy while avoiding significant runtime overhead.

11.3. Hierarchical Sampling

Deep learning-based shape retrieval methods usually involve calculating an embedding for the input shape, and retrieval is done by comparing the cosine similarity between the embeddings. However, as seen in Tab. 18, these methods do not perform well on the Category-Chamfer metric. Unlike cosine similarity which can be easily computed in a batched manner, Chamfer Distance requires comparison between individual point clouds. A naive solution is to calculate the Chamfer Distance with all other shapes within the same category. However, such a naive method would require extensive computation, scaling linearly with the size of the dataset for retrieval.

To this end, we propose a Hierarchical Sampling approach, visualized in Fig. 8. We found that the Chamfer Distance at a small number of points (128) is an effective proxy for the Chamfer Distance at a large number of points (4096). Although we calculate the groundtruth Chamfer

Method	Ours		PointNeXt
Input Type	NGLOD	iNGP	Point Cloud
A_C	82.6	**84.2**	71.2
A_{CC}	21.2	23.2	**28.4**

Table 18. Retrieval Accuracy (Category, Category-Chamfer) on ShapeNet10

Method	Ours				PointNeXt	
Input Type	NGLOD		iNGP		Point Cloud	
A_{CC}	81.8		82.4		72.6	
Ret. Time (Naive)	65.06		65.06		65.06	
Ret. Time (Hier. Samp.) Total	36.19		35.46		35.78	
Ret. Time (Hier. Samp.) CD@128/4096	25.05	11.14	25.05	10.41	25.05	10.73

Table 19. Category-Chamfer Retrieval Accuracy and Retrieval Time on ShapeNet10

Distance at 131072 points following typical values used for evaluation of 3D shape reconstruction quality [42], we found that in terms of ranking of shape by Chamfer Distance, 4096 points is sufficient. For INRet, we use the frozen INR Embeddings to train an MLP for classification, following the same settings as [13]. We use E-Stitchup to augment the input with interpolations of INR embeddings from the same class [49]. For PointNeXt, we use the trained PointNeXt to do the classification.

We present the result of the retrieval in Tab. 19. For naive retrieval, we directly sample points and calculate the Chamfer Distance at 4096 points between the query INR and all candidate INRs. For Hierarchical Sampling retrieval, we first sample points and calculate the Chamfer Distance at 128 points between the query INR and all candidate INRs. We further calculate the Chamfer Distance at 4096 points for all INRs with a small Chamfer Distance at 128 points. We define small by the INR having Chamfer Distance within 3 times of the smallest Chamfer Distance between query INR and all candidate INRs. This is a very generous bound and ensures a 100% recall on our dataset. The accuracy is effectively only limited by the classification accuracy.

As shown in Tab. 19, using Hierarchical Sampling significantly reduces the time (on average 1.8X) required for calculating the Chamfer Distance between different INRs. The speed-up for all methods is very similar as the point sampling and Chamfer Distance calculation time dominates the runtime. This leaves the difference in time for classification between the methods negligible. Using NGLOD as an example, the naive retrieval method involves point sampling and Chamfer Distance calculation (4096 points) for 49 INRs which costs 65.06 seconds, and an additional 0.04 seconds for classification. Using the hierarchical method, the distance point sampling and Chamfer Distance calculation are first done for 128 points (25.05 seconds + 0.04 seconds for classification), and around 17.1% of the INRs need to be further evaluated at 4096 points, resulting in a runtime of 11.14 seconds. We expect this speedup to scale further as more data is presented as the retrieval candidate. Despite the speed up, this process is still relatively slow compared to the category-only retrieval which typically only requires cosine similarity comparison. We leave potential methods that would allow fast and accurate Category-Chamfer retrieval as future work.