Supplementary

In the following sections, we provide additional details and results of our sampling approach. Section A presents additional results of our method. An ablation study is reported in Section B. Section C describes mathematical aspects of the soft projection operation, employed by SampleNet. Finally, experimental settings, including network architecture and hyperparameter settings, are given in Section D.

A. Additional results

A.1. Point cloud retrieval

We employ sampled point sets for point cloud retrieval, using either farthest point sampling (FPS), S-NET, or SampleNet. In order to evaluate cross-task usability, the last two sampling methods are trained with PointNet for classification and applied for the retrieval task without retraining [6]. The shape descriptor is the activation vector of the second-last layer of PointNet when it fed with sampled or complete clouds. The distance metric is l_2 between shape descriptors.

Precision and recall are evaluated on the test set of ModelNet40, where each shape is used as a query. The results when using the complete 1024 point sets and samples of 32 points are presented in Figure 14. SampleNet improves the precision over all the recall range with respect to S-NET and approaches the performance with complete input sets. It shows that the points sampled by SampleNet are suitable not only for point cloud classification but also for retrieval.

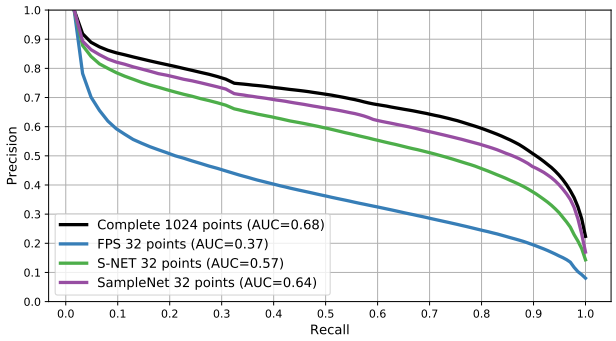

Figure 14. **Precision-recall curve with sampled points.** PointNet is fed with sampled point clouds from the test set. Its penultimate layer is used as the shape descriptor. Utilizing SampleNet results in improved retrieval performance in comparison to the other sampling methods. Using only 32 points, SampleNet is close to the precision obtained with complete input points cloud, with a drop of only 4% in the area under the curve (AUC).

A.2. Progressive sampling

Our method is applied to the progressive sampling of point clouds [6] for the classification task. In this case, the vanilla version of PointNet [28] is employed as the classifier [6]. Performance gains are achieved in the progressive sampling settings, as shown in Figure 15. They are smaller than those of SampleNet trained per sample size separately (see Figure 5 in the main body) since for progressive sampling, SampleNet-Progressive should be optimal for all the control sizes concurrently.

We also perform reconstruction from progressively sampled point clouds. Our normalized reconstruction error is compared to that of FPS and ProgressiveNet [6] in Figure 16. Figure 21 shows a visual reconstruction example.

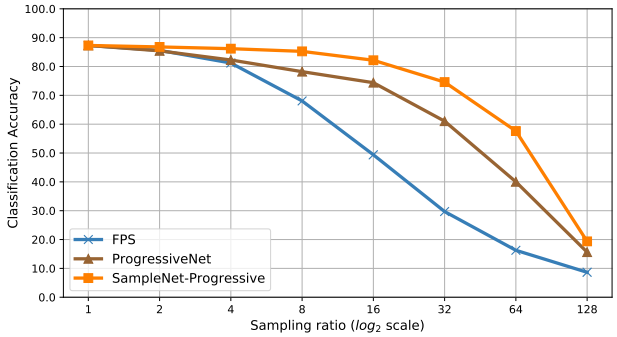

Figure 15. **Classification results with SampleNet-Progressive.** PointNet vanilla is used as the task network and was pre-trained on point clouds with 1024 points. The instance classification accuracy is evaluated on sampled point clouds from the test split. Our sampling network outperforms farthest point sampling (FPS) and ProgressiveNet [6].

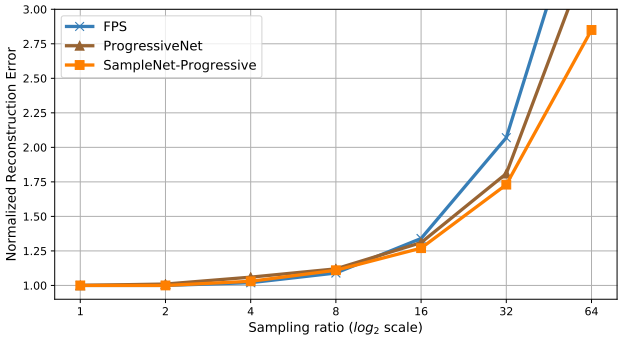

Figure 16. **Normalized reconstruction error with SampleNet-Progressive.** Point clouds are reconstructed from nested sets of sampled points. We normalize the reconstruction error from a sample by the error resulting from a complete input. As the sampling ratio is increased, the improvement of SampleNet, compared to the alternatives, becomes more dominant.

A.3. Computation load and memory space

The computation load of processing a point cloud through a network is regarded as the number of multiply-accumulate operations (MACs) for inference. The required

memory space is the number of learnable parameters of the network.

For a PointNet like architecture, the number of MACs is mainly determined by the number of input points processed by the multi-layer perceptrons (MLPs). Thus, reducing the number of points reduces the computational load. The memory space of SampleNet depends on the number of output points, resulting from the last fully connected layer. The soft projection operation adds only one learnable parameter, which is negligible to the number of weights of SampleNet.

We evaluate the computation load and memory space for the classification application. We denote the computation and memory of SampleNet that outputs m points as C_{SN_m} and M_{SN_m}, respectively. Similarly, the computation of PointNet that operates on m points is denoted as C_{PN_m}, and for a complete point cloud as C_{PN}. The memory of PointNet is marked M_{PN}. It is independent of the number of processed points. When concatenating SampleNet with PointNet, we define the computation reduction percent CR as:

$$CR = 100 \cdot \left(1 - \frac{C_{SN_m} + C_{PN_m}}{C_{PN}}\right), \tag{13}$$

and the memory increase percent MI as:

$$MI = 100 \cdot \frac{M_{SN_m} + M_{PN}}{M_{PN}}. \tag{14}$$

Figure 17 presents the memory increase versus computation reduction. As the number of sampled points is reduced, the memory increase is lower, and the computation reduction is higher, with a mild decrease in the classification accuracy.

For example, SampleNet for 32 points has 0.22M parameters and performs 34M MACs ('M' stands for Million). PointNet that operates on point clouds of 32 instead of 1024 points requires only 14M instead of 440M MACs. The number of PointNet parameters is 3.5M. SampleNet followed by PointNet sums up to 48M MACs and 3.72M parameters. These settings require about 6% additional memory space and reduce the computational load by almost 90%.

A.4. Sampling consistency for registration task

Given a sampled set T_s^{gt} of template points, rotated by the ground truth rotation R_{gt}, and a sampled set S_s of source points, the sampling consistency is defined as the Chamfer distance between these two sets:

$$\begin{aligned} C(S_s, T_s^{gt}) = &\frac{1}{|S_s|} \sum_{\mathbf{t} \in S_s} \min_{\mathbf{t} \in T_s^{gt}} ||\mathbf{s} - \mathbf{t}||_2^2 \\ &+ \frac{1}{|T_s^{gt}|} \sum_{\mathbf{t} \in T_s^{gt}} \min_{\mathbf{s} \in S_s} ||\mathbf{t} - \mathbf{s}||_2^2. \end{aligned} \tag{15}$$

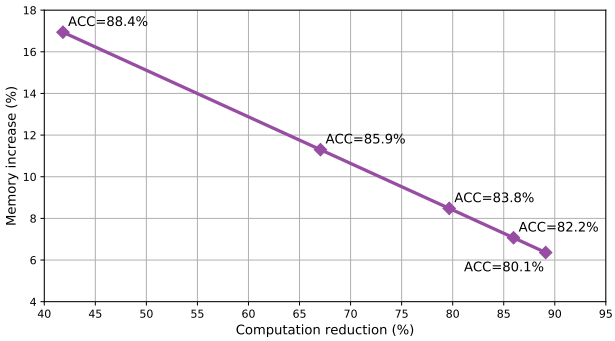

Figure 17. **Memory, computation, and performance.** The memory increase for chaining SampleNet with PointNet is plotted against the computation reduction, which results from processing sampled instead of complete clouds. The points on the graph from left to right correspond to sampling ratios $\{2, 4, 8, 16, 32\}$. ACC is the classification accuracy on the test split of ModelNet40 when PointNet runs on sampled point sets. With a slight increase in memory and small accuracy drop, SampleNet reduces the computational load substantially.

For a given sampler, this metric quantifies the tendency of the algorithm to sample similar points from the source and template point clouds. We measure the average consistency on the test set of the Car category from ModelNet40. Results for random sampling, FPS and SampleNet are reported in Table 2. The table shows that SampleNet sampling is substantially more consistent than that of the alternatives. This behavior can explain its success for the registration task.

A.5. Registration for different shape categories

Registration is applied to different shape categories from ModelNet40. We present the results for Table, Sofa, and Toilet categories in Table 3, and visualizations in Figure 18. Additional shape classes that we evaluated include Chair, Laptop, Airplane and Guitar. SampleNet achieves the best results compared to FPS and random sampling for all these categories.

B. Ablation study

B.1. Neighborhood size

The neighborhood size $k = |\mathcal{N}_P(\mathbf{q})|$ is the number of neighbors in P of a point $\mathbf{q} \in Q$, on which $\mathbf{q}$ is softly projected. This parameter controls the local context in which $\mathbf{q}$ searches for an optimal point to sample.

We assess the influence of this parameter by training several progressive samplers for classification with varying values of k. Figure 19 presents the classification accuracy difference between SampleNet-Progressive trained with $k = 7$ and with $k \in \{2, 4, 12, 16\}$. The case of $k = 7$ serves as a baseline, and its accuracy difference is set to 0. As shown

Sampling ratio	2	4	8	16	32	64	128
Random sampling	1.03	2.59	5.29	9.99	18.53	34.71	63.09
FPS	**0.46**	**1.5**	3.3	6.42	11.78	22.23	43.49
SampleNet	0.53	1.64	**3.14**	**4.83**	**6.85**	**7.2**	**9.6**

Table 2. **Sampling consistency between rotated point clouds.** The consistency is measured for the test split of Car category from ModelNet40. The results are multiplied by a factor of 10^3. Lower is better. When the sampling ratio is small and many points are taken, SampleNet performs on par with the other methods. However, as it increases, SampleNet selects much more similar points than random sampling and FPS.

Category	*Table*			*Sofa*			*Toilet*		
Sampling ratio	8	16	32	8	16	32	8	16	32
Random sampling	13.09	18.99	29.76	16.58	24.57	34.19	12.17	20.51	35.92
FPS	7.74	8.79	11.15	9.41	12.13	17.52	7.74	8.49	11.69
SampleNet	**6.44**	**7.24**	**8.35**	**8.56**	**10.8**	**10.97**	**6.05**	**7.09**	**8.07**

Table 3. **Mean rotation error (MRE) with SampleNet for different shape categories.** MRE is reported in degrees. Lower is better. PCRNet is trained on complete point clouds of 1024 points from the Table, Sofa and Toilet categories of ModelNet40. The MRE is measured on the test split for different sampling methods. Utilizing SampleNet yields better results. With complete input, PCRNet achieves 6.08° MRE for Table, 7.15° MRE for Sofa, and 5.43° MRE for Toilet.

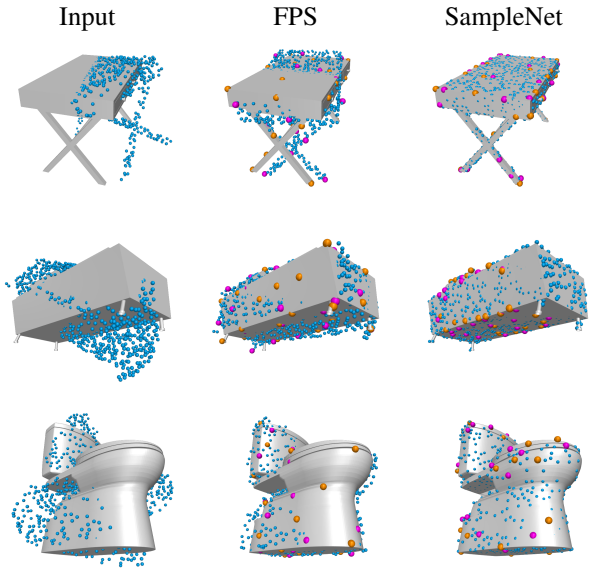

Figure 18. **Registration with sampled points for different shape categories.** Left column: unregistered source with 1024 points in Blue overlaid on the mesh model. Middle column: FPS registered results. Right column: SampleNet registered results. Sampled sets of 32 points from the template and source are illustrated in Orange and Magenta, respectively. Registration with SampleNet points yields better results than FPS.

in the figure, training with smaller or larger neighborhood sizes than the baseline decreases the accuracy. We conclude that $k = 7$ is a sweet spot in terms of local exploration region size for our learned sampling scheme.

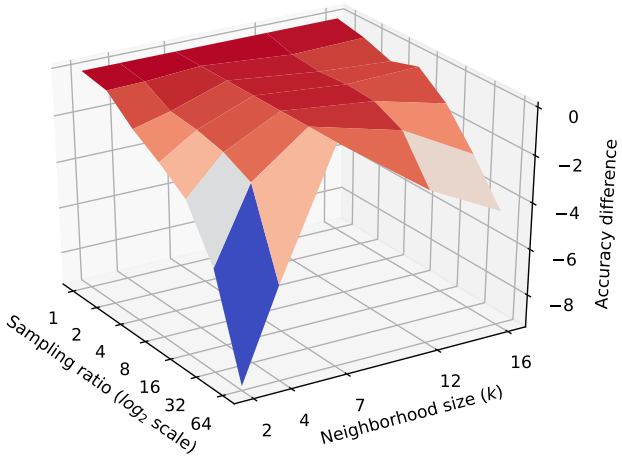

Figure 19. **The influence of different neighborhood sizes.** SampleNet-Progressive is trained for classification with different sizes k for the projection neighborhood and evaluated on the test split of ModelNet40. We measure the accuracy difference for each sampling ratio with respect to the baseline of $k = 7$. Larger or smaller values of k result in negative accuracy difference, which indicates lower accuracy.

B.2. Additional loss terms

As noted in the paper in section 4.1, the average soft projection weights, evaluated on the test set of ModelNet40, are different than a delta function (see Figure 7). In this experiment, we examine two loss terms, cross-entropy and entropy loss, that encourage the weight distribution to converge to a delta function.

For a point $\mathbf{q} \in Q$, we compute the cross-entropy between a Kronecker delta function, representing the nearest

neighbor of $\mathbf{q}$ in P, and the projection weights of $\mathbf{q}$, namely, $\{w_i\}$, $i \in \mathcal{N}_P(\mathbf{q})$. The cross-entropy term takes the form:

$$H_P^c(\mathbf{q}) = -\sum_{i \in \mathcal{N}_P(\mathbf{q})} \mathbb{1}_{i^*}(i) log(w_i) = -log(w_{i^*}), \quad (16)$$

where $\mathbb{1}_{i^*}(i)$ is an indicator function that equals 1 if $i = i^*$ and 0 otherwise; $i^* \in \mathcal{N}_P(\mathbf{q})$ is the index of nearest neighbor of $\mathbf{q}$ in P. The cross-entropy loss is the average over all the points in Q:

$$\mathcal{L}_c(Q, P) = \frac{1}{|Q|} \sum_{\mathbf{q} \in Q} H_P^c(\mathbf{q}). \quad (17)$$

Similarly, the entropy of the projection weights for a point $\mathbf{q} \in Q$ is given by:

$$H_P(\mathbf{q}) = -\sum_{i \in \mathcal{N}_P(\mathbf{q})} w_i log(w_i), \quad (18)$$

and the entropy loss is defined as:

$$\mathcal{L}_h(Q, P) = \frac{1}{|Q|} \sum_{\mathbf{q} \in Q} H_P(\mathbf{q}). \quad (19)$$

The cross-entropy and entropy losses are minimized when one of the weights is close to 1, and the others to 0. We add either of these loss terms, multiplied by a factor η, to the training objection of SampleNet (Equation 1), and train it for the classification task.

Figure 20 presents the weight evolution for SampleNet that samples 64 points. It was trained with the additional cross-entropy loss, with $\eta = 0.1$. In these settings, the weights do converge quite quickly to approximately delta function, with an average weight of 0.94 for the first nearest neighbor at the last epoch. However, as Table 4 shows, this behavior does not improve the task performance, but rather the opposite.

The cross-entropy loss compromises the quest of SampleNet for optimal points for the task. Instead of exploring their local neighborhood, the softly projected points are locked on their nearest neighbor in the input point cloud early in the training process. We observed similar behavior when using the entropy loss instead of the cross-entropy loss. We conclude that the exact convergence to the nearest neighbor is not required. Instead, the projection loss (Equation 10) is sufficient for SampleNet to achieve its goal - learning to sample an optimal point set for the task at hand.

C. Mathematical aspects of soft projection

C.1. Idempotence

Idempotence is a property of an operation whereby it can be applied several times without changing the obtained initial result. A mathematical projection is an idempotent operation. In the limit of $t \to 0$, the soft projection becomes

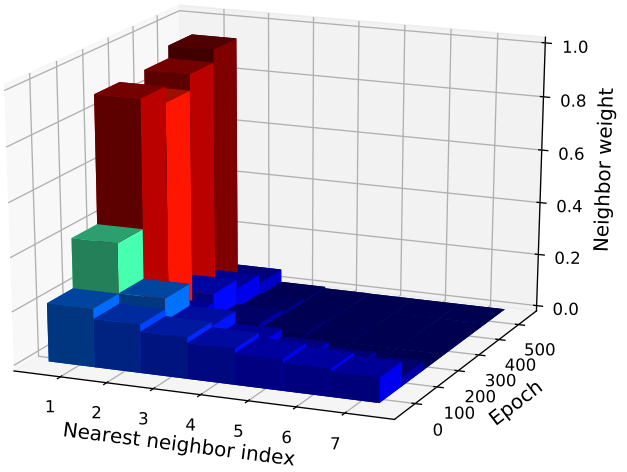

Figure 20. **Weight evolution with cross-entropy loss.** SampleNet is trained to sample 64 points for classification. A cross-entropy loss on the projection weights is added to its objective function. The weights are averaged on sampled point clouds from the test set of ModelNet40 after the first and every 100 training epochs. In these settings, most of the weight is given to the first nearest neighbor quite early in the training process.

an idempotent operation. That is:

$$\lim_{t \to 0} \sum_{i \in \mathcal{N}_P(\mathbf{q})} w_i(t) \mathbf{p}_i = \underset{\{\mathbf{p}_i\}}{\operatorname{argmin}} ||\mathbf{q} - \mathbf{p}_i||_2 = \mathbf{r}^*, \quad (20)$$

which results in the definition of sampling in Equation 12. The proof of idempotence for the sampling operation is straightforward:

$$\underset{\{\mathbf{p}_i\}}{\operatorname{argmin}} ||\mathbf{r}^* - \mathbf{p}_i||_2 = \mathbf{r}^*. \quad (21)$$

C.2. Projection under the Bregman divergence

The distance we choose to minimize between a query point $\mathbf{q} \in Q$ and the initial point cloud P is the Squared Euclidean Distance (SED). However, SED is not a metric; it does not satisfy the triangle inequality. Nevertheless, it can be viewed as a Bregman divergence [4], a measure of distance defined in terms of a convex generator function F.

Let $F : X \to \mathbb{R}$ be a continuously-differentiable and convex function, defined on a closed convex set X. The Bregman divergence is defined to be:

$$D_F(\mathbf{p}, \mathbf{q}) = F(\mathbf{p}) - F(\mathbf{q}) - \langle \nabla F(\mathbf{q}), \mathbf{p} - \mathbf{q} \rangle. \quad (22)$$

Choosing $F(\mathbf{x}) : \mathbb{R}^k \to \mathbb{R} = \|\mathbf{x}\|^2$, the Bregman divergence takes the form:

$$D_F(\mathbf{p}, \mathbf{q}) = \|\mathbf{p} - \mathbf{q}\|^2. \quad (23)$$

The projection under the Bregman divergence is defined as follows. Let $\zeta \subseteq \mathbb{R}^k$ be a closed, convex set. Assume

Sampling ratio	2	4	8	16	32	64	128
SampleNet trained with cross entropy loss	88.2	83.4	79.7	79.0	74.4	**55.5**	**28.7**
SampleNet trained without cross entropy loss	**88.4**	**85.9**	**83.8**	**82.2**	**80.1**	54.0	23.2

Table 4. **Ablation test for cross-entropy loss.** SampleNet is trained for classification, either with or without cross-entropy loss (Equation 17). For each case, we report the classification accuracy on the test split of ModelNet40. Employing cross-entropy loss during training results in inferior performance for most of the sampling ratios.

that $F : \zeta \to \mathbb{R}$ is a strictly convex function. The projection of $\mathbf{q}$ onto ζ under the Bregman divergence is:

$$\Pi_{\zeta}^{F}(\mathbf{q}) \triangleq \underset{\mathbf{r}\in\zeta}{\operatorname{argmin}} D_F(\mathbf{r}, \mathbf{q}). \tag{24}$$

In our settings, the softly projected points are a subset of the convex hull of $\{\mathbf{p}_i\}$, $i \in \mathcal{N}_P(\mathbf{q})$. The convex hull is a closed and convex set denoted by $\zeta_{\mathbf{q}}$:

$$\zeta_{\mathbf{q}} = \left\{ \mathbf{r} : \mathbf{r} = \sum_{i\in\mathcal{N}_P(\mathbf{q})} w_i \mathbf{p}_i, w_i \in [0,1], \sum_{i\in\mathcal{N}_P(\mathbf{q})} w_i = 1 \right\} \tag{25}$$

In general, not all the points in $\zeta_{\mathbf{q}}$ can be obtained, because of the restriction imposed by the definition of $\{w_i\}$ in Equation 9. However, as we approach the limit of $t \to 0$, the set $\zeta_{\mathbf{q}}$ collapses to $\{\mathbf{p}_i\}$. Thus, we obtain the sampling operation:

$$\Pi_{\mathcal{N}_P(\mathbf{q})}^{F}(\mathbf{q}) \triangleq \underset{\{\mathbf{p}_i\}}{\operatorname{argmin}} D_F(\mathbf{p}_i, \mathbf{q}) = \mathbf{r}^*, \tag{26}$$

as defined in Equation 12.

D. Experimental settings

D.1. Task networks

We adopt the published architecture of the task networks, namely, PointNet for classification [28], PCRNet for registration [32], and point cloud autoencoder (PCAE) for reconstruction [1]. PointNet and PCAE are trained with the settings reported by the authors. Sarode *et al.* [32] trained PCRNet with Chamfer loss between the template and registered point cloud. We also added a loss term between the estimated transformation and the ground truth one. We found out that this additional loss term improved the results of PCRNet, and in turn, the registration performance with sampled point clouds of SampleNet. Section D.4 describes both loss terms.

D.2. SampleNet architecture

SampleNet includes per-point convolution layers, followed by symmetric global pooling operation and several fully connected layers. Its architecture for different applications is detailed in Table 5. For SampleNet-Progressive, the architecture is the same as the one in the table, with $m = 1024$ for classification and $m = 2048$ for reconstruction.

Task	SampleNet architecture
Classification	$MLP(64, 64, 64, 128, 128)$
	max pooling
	$FC(256, 256, 256, m \times 3)$
Registration	$MLP(64, 64, 64, 128, 128)$
	max pooling
	$FC(256, 256, 256, m \times 3)$
Reconstruction	$MLP(64, 128, 128, 256, 128)$
	max pooling
	$FC(256, 256, m \times 3)$

Table 5. **SampleNet architecture for different tasks.** MLP stands for multi-layer perceptrons. FC stands for fully connected layers. The values in $MLP(\cdot)$ are the number of filters of the per-point convolution layers. The values in $FC(\cdot)$ are the number of neurons of the fully connected layers. The parameter m in the last fully connected layer is the sample size.

Each convolution layer includes batch normalization and ReLU non-linearity. For classification and registration, each fully connected layer, except the last one, includes batch normalization and ReLU operations. ReLU is also applied to the first two fully connected layers for the reconstruction task, without batch normalization.

D.3. SampleNet optimization

Table 6 presents the hyperparameters for the optimization of SampleNet. In progressive sampling for the classification task, we set $\gamma = 0.5$ and $\delta = 1/30$. The other parameter values are the same as those appear in the table. We use Adam optimizer with a momentum of 0.9. For classification, the learning rate decays by a factor of 0.7 every 60 epochs. SampleNet-Progressive is trained with control sizes $C_s = \{2^l\}_{l=1}^{10}$ for classification and $C_s = \{2^l\}_{l=4}^{12}$ for reconstruction.

The temperature coefficient (t in Equation 9) is initialized to 1 and learned during training. In order to avoid numerical instability, it is clipped by a minimum value of 0.1 for registration and 0.01 for reconstruction.

We train our sampling method with a Titan Xp GPU. Training SampleNet for classification takes between 1.5 to 7 hours, depending on the sample size. The training time

	Classification	Registration	Reconstruction
k	7	8	16
α	30	0.01	0.01
β	1	1	1
γ	1	1	0
δ	0	0	1/64
λ	1	0.01	0.0001
BS	32	32	50
LR	0.01	0.001	0.0005
TEs	500	400	400

Table 6. **Hyperparameters.** The table details the values that we use for the training of our sampling method for different applications. BS, LR, and TEs stand for batch size, learning rate, and training epochs, respectively.

of progressive sampling for this task is about 11 hours. The training time of SampleNet for registration takes between 1 to 2.5 hours. For the sample sizes of the reconstruction task, SampleNet requires between 4 to 30 hours of training, and SampleNet-Progressive requires about 2.5 days.

D.4. Losses and evaluation metric for registration

Since the code of PCRNet [32] was unavailable at the time of submission, we trained PCRNet with slightly different settings than those described in the paper, by using a mixture of supervised and unsupervised losses.

The unsupervised loss is the Chamfer distance [1]:

$$\begin{aligned}\mathcal{L}_{cd}(S,T) = \frac{1}{|S|}\sum_{\mathbf{s}\in S}\min_{\mathbf{t}\in T}||\mathbf{s}-\mathbf{t}||_2^2 \\ +\frac{1}{|T|}\sum_{\mathbf{t}\in T}\min_{\mathbf{s}\in S}||\mathbf{t}-\mathbf{s}||_2^2,\end{aligned} \tag{27}$$

for a source point cloud S and a template point cloud T. For the supervised loss, we take the quaternion output of PCRNet and convert it to a rotation matrix to obtain the predicted rotation R_{pred}. For a ground truth rotation R_{gt}, the supervised loss is defined as follows:

$$\mathcal{L}_{rm}(R_{pred}, R_{gt}) = ||R_{pred}^{-1}\cdot R_{gt} - I||_F^2, \tag{28}$$

where I is a 3×3 identity matrix, and $||\cdot||_F$ is the Frobenius norm. In total, the task loss for registration is given by $\mathcal{L}_{cd}(S,T) + \mathcal{L}_{rm}(R_{pred}, R_{gt})$.

The rotation error RE is calculated as follows [53]:

$$RE = 2cos^{-1}(2\langle q_{pred}, q_{gt}\rangle^2 - 1), \tag{29}$$

where q_{pred} and q_{gt} are quaternions, representing the predicted and ground truth rotations, respectively. We convert the obtained value from radians to degrees, average over the test set, and report the mean rotation error.

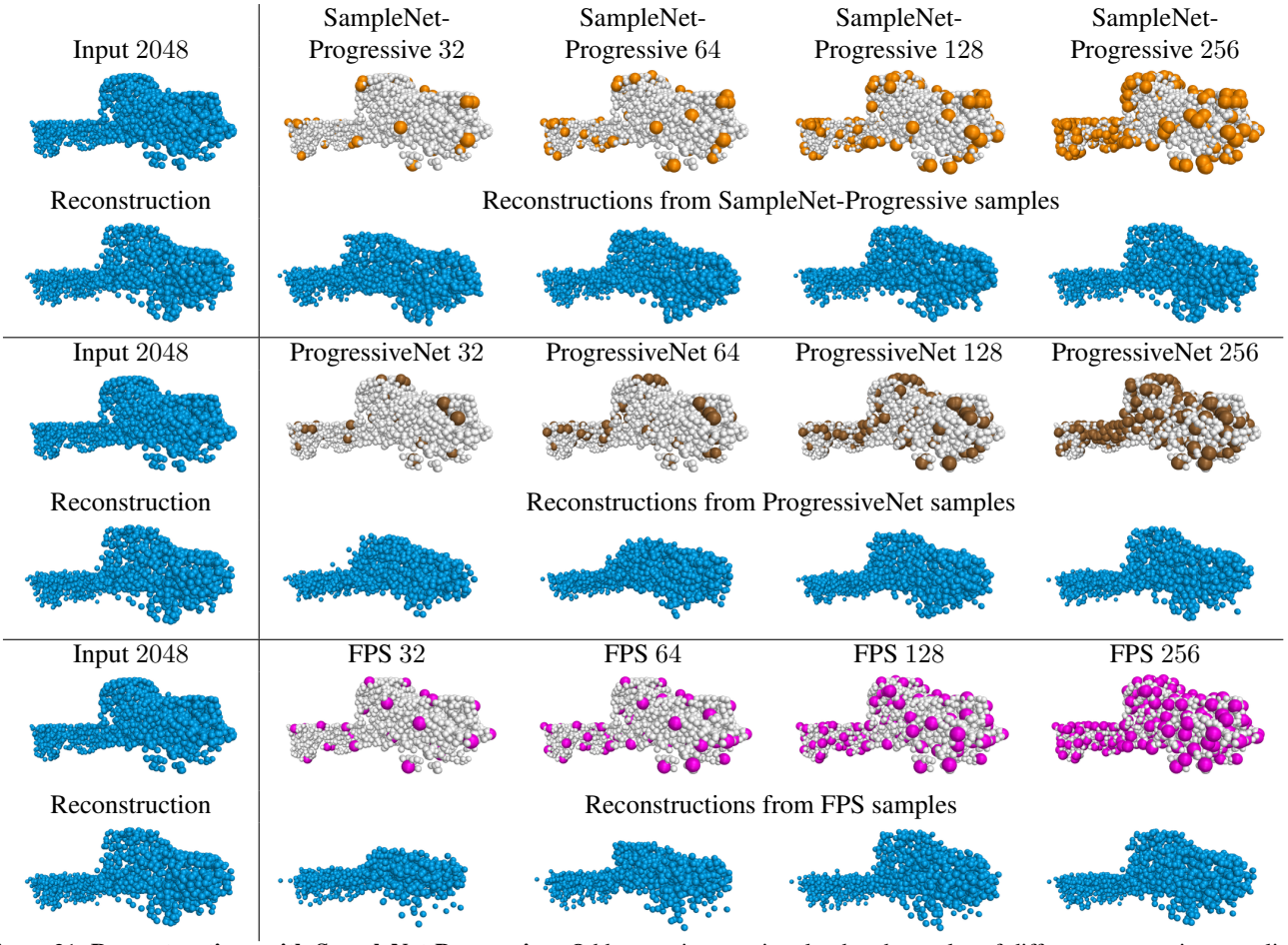

Figure 21. **Reconstructions with SampleNet-Progressive.** Odd rows: input point cloud and samples of different progressive sampling methods. The number of sampled points is denoted next to the method's name. Even rows: reconstruction from the input and the corresponding sample. Our SampleNet-Progressive selects most of its points at the outline of the shape, while ProgressiveNet [6] selects interior points and FPS points are spread uniformly. In contrast to the other methods, our result starts to resemble the reconstruction from the complete input when using only 32 points, which is about 1.5% of the input data.