

A Proof of Proposition 1

Proposition 1. *Jointly optimizing the synthetic dataset \mathcal{S} and the rotation parameters $\theta = (\theta_x, \theta_y, \theta_z)$ guarantees a lower or equal loss to that of optimizing \mathcal{S} alone.*

$$\min_{\mathcal{S}, \theta} \mathcal{L}_{\text{SADM}}(\mathcal{T}, \mathcal{R}_\theta(\mathcal{S})) \leq \min_{\mathcal{S}} \mathcal{L}_{\text{SADM}}(\mathcal{T}, \mathcal{S}), \quad (1)$$

where \mathcal{R}_θ denotes the rotation operator according to θ .

Proof. Let us consider two optimization objectives. The first is the baseline optimization, which optimizes only the synthetic dataset without applying any rotation

$$L := \min_{\mathcal{S}} \mathcal{L}_{\text{SADM}}(\mathcal{T}, \mathcal{S}). \quad (2)$$

The second is the proposed joint optimization over both the synthetic dataset \mathcal{S} and the rotation parameters $\theta = (\theta_x, \theta_y, \theta_z)$, given

$$L^* := \min_{\mathcal{S}, \theta} \mathcal{L}_{\text{SADM}}(\mathcal{T}, \mathcal{R}_\theta(\mathcal{S})). \quad (3)$$

Consider a special case of the proposed joint optimization framework where no rotation is applied, i.e., the rotation parameters are fixed to $\theta = (0, 0, 0)$. In such a case, $\mathcal{R}_\theta(\mathcal{S}) = \mathcal{S}$, and the baseline optimization can be regarded as a special case of the proposed joint optimization. This implies that the feasible set of the baseline optimization is a subset of that of the joint optimization such that

$$\{(\mathcal{S}, \theta) \mid \theta = (0, 0, 0)\} \subseteq \{(\mathcal{S}, \theta)\}. \quad (4)$$

The joint optimization is performed over a larger feasible set than that of the baseline optimization, and therefore its optimal solution must not be higher than that of the baseline. Therefore,

$$L^* \leq L. \quad (5)$$

B Experimental Details

Datasets. Table 1 summarizes the number of training and testing samples for ModelNet10 [11], ModelNet40 [11], ShapeNet [2], and ScanObjectNN [8].

- **ModelNet10** consists of 10 categories of aligned 3D CAD models. There is no rotational variation.
- **ModelNet40** includes 40 categories from the same source to ModelNet10, with partially misaligned instances introducing moderate rotational variation.
- **ShapeNet** contains 55 categories of manually aligned 3D models without rotational variation.
- **ScanObjectNN** consists of 15 categories from real-world RGB-D scans with background clutter, occlusion, and sensor noise. Objects are unaligned, and rotation variation is inherent. We use the **PB_T50_RS** split, which includes perturbed background, translation jitter, rotation, and scaling.

Table 1: Train/test statistics for ModelNet10, ModelNet40, ShapeNet and ScanObjectNN.

	ModelNet10	ModelNet40	ShapeNet	ScanObjectNN
# of training samples	3991	9843	35708	11416
# of test samples	908	2468	10261	2882
# of classes	10	40	55	15
# of points for each sample	1024	1024	1024	1024

Network Architecture. We evaluate our method on five representative point-based architectures: PointNet [4], PointNet++ [5], DGCNN [9], PointConv [10], and Point Transformer [14].

- **PointNet** is a widely used neural network designed for processing 3D point clouds. We use PointNet as the backbone in dataset distillation model. While the original PointNet includes two transformation modules for aligning input and features, we only use the input transformation module. The rest of the architecture follows the standard PointNet design.
- **PointNet++** is used for cross-architecture evaluation. To reduce computational cost and accelerate training, we decrease the width of all MLP layers by half, while keeping the overall structure unchanged. We use two set abstraction modules with multi-scale grouping and one with single-scale grouping to extract point features, followed by two linear layers and a final classifier for prediction.
- **DGCNN** employs dynamic graph construction and EdgeConv layers to capture local geometric relationships by recomputing a k -NN graph ($k = 20$) at each layer in the feature space. We use the default architecture, which consists of four EdgeConv layers followed by two linear layers and a final classifier.
- **PointConv** extends convolution to point clouds by accounting for non-uniform point densities during feature learning. The architecture consists of three density-aware set abstraction layers, followed by two linear layers and a final classifier.
- **Point Transformer** applies self-attention mechanisms to point clouds to capture both local and global dependencies. We use the default configuration with four transformer blocks, followed by two linear layers and a final classifier.

Implementation Details. We optimized the synthetic dataset \mathcal{S} using stochastic gradient descent (SGD) with a learning rate of 0.01, a momentum of 0.5, a weight decay of 0, and a batch size of 8 per class sampled from the original dataset \mathcal{T} , while the batch size of the synthetic dataset was set equal to the number of synthetic samples per class (PPC). The synthetic dataset optimization was performed for 1,500 iterations, and the corresponding configuration is summarized in Table 2a. To preserve fine-grained geometric details, the SADMM loss was computed using the feature maps before the max pooling layer. Additionally, a secondary loss term was applied to the top-1 sorted feature values in each channel to emphasize semantically dominant regions. The loss weights λ_1 and λ_2 were determined based on PPC, and set to 0.002, 0.006, and 0.02 for λ_1 , and 0.001, 0.003, and 0.01 for λ_2 when PPC was 1, 3, and 10, respectively.

The rotation parameters $\theta = (\theta_x, \theta_y, \theta_z)$ were jointly optimized with the synthetic dataset using SGD with a momentum of 0.5 and a weight decay of 0. The learning rates were set to 0.5 for θ_x and θ_z , and 5.0 for θ_y , reflecting that the samples in datasets are vertically aligned. A step decay scheduler was used with a step size of 100 and a decay factor of 0.5. This setup is detailed in Table 2b.

For evaluation, we trained all backbone networks including PointNet, PointNet++, DGCNN, PointConv, and Point Transformer using SGD with a learning rate of 0.01, a momentum of 0.9, and a weight decay of 0.0005. A step decay scheduler was applied with a step size of 250 and a decay factor of 0.1. Each model was trained for 500 epochs with a batch size of 8. These test-time settings are listed in Table 2c.

All experiments were conducted on a single NVIDIA GeForce RTX 3090 GPU.

Table 2: Hyperparameter settings used for (a) optimizing the synthetic dataset, (b) optimizing the rotation parameters, and (c) evaluation network.

Hyperparameters	PPC1	PPC3	PPC10
Optimizer	SGD	SGD	SGD
Momentum	0.5	0.5	0.5
Weight Decay	0.0	0.0	0.0
Learning Rate	0.01	0.01	0.01
Iteration	1500	1500	1500
λ_1	0.002	0.006	0.02
λ_2	0.001	0.003	0.01
Batch Size (\mathcal{T})	8	8	8

(a)

Hyperparameters	
Optimizer	SGD
Momentum	0.5
Weight Decay	0.0
Learning Rate (θ_x)	0.5
Learning Rate (θ_y)	5.0
Learning Rate (θ_z)	0.5
Scheduler	StepLR
Step Size	100
Gamma	0.5

(b)

Hyperparameters	
Optimizer	SGD
Momentum	0.9
Weight Decay	0.0005
Learning Rate	0.01
Epochs	500
Batch Size	8
Scheduler	StepLR
Step Size	250
Gamma	0.1

(c)

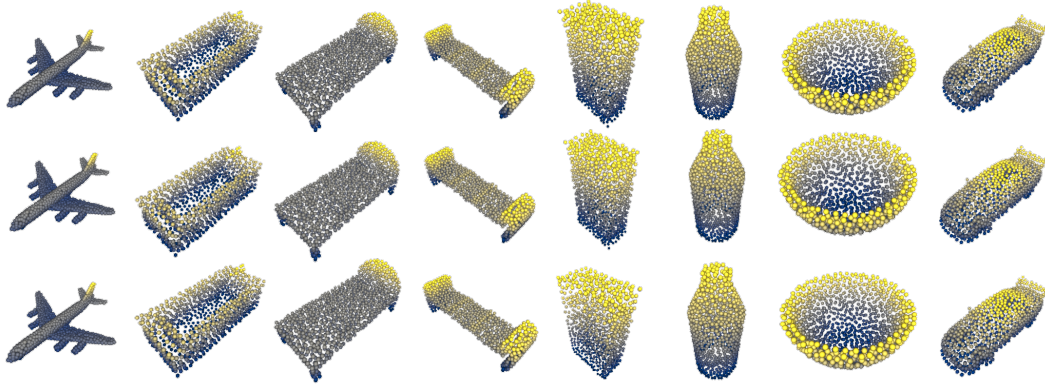


Figure 1: Visualization of synthetic samples of ModelNet40 [11] at PPC 1, obtained by using the random selection [6](top), DataDAM [7](middle), and MTT [1](bottom).

Baselines. All baseline methods were implemented on 3D point clouds using PointNet as the feature extractor, following their original designs. All methods were trained for 1,500 iterations using a batch size of 8 per class from the original dataset \mathcal{T} , and the training configurations are summarized in Table 3. Details for each method are as follows

- **DC** [13]: We followed the original framework and performed gradient matching. The number of inner and outer loop steps were set to 1/1/10 and 1/5/3 for PPC values of 1, 3, and 10, respectively
- **DM** [12]: Following the original setup, we used the feature vector obtained before the final classifier to compute the matching loss between the original and synthetic datasets.
- **MTT** [1]: We followed the original framework for trajectory matching, with modifications only in the hyperparameter settings.
- **DataDAM** [7]: We followed the original framework for feature matching, making only minor adjustments to hyperparameters.

Table 3: Hyperparameter settings of the baselines.

	DC	DM	MTT	DataDAM
Mode	PointNet	PointNet	PointNet	PointNet
Learning Rate	0.0001	1	0.0001	0.0001
Batch Size (\mathcal{T})	8	8	8	8
Iteration	1500	1500	1500	1500
Inner Loop	1 / 1 / 10	—	—	—
Outer Loop	1 / 5 / 3	—	—	—

C Comparison with Additional Baselines

Table 4 presents the additional comparison results of the classification accuracy with DataDAM [7] and MTT [1]. For DataDAM and MTT, we used a possible learning rate to prevent divergence on the synthetic datasets. The performance of both DataDAM and MTT is almost similar to that of random selection [6], as shown in the second and third rows in Figure 1 where most points remain nearly unchanged from their random initialization in the first row.

Table 4: Classification accuracy of the proposed method compared with DataDAM [7] and MTT [1] initialized with randomly selected samples.

Datasets	ModelNet10			ModelNet40			ShapeNet			ScanObjectNN		
PPC	1	3	10	1	3	10	1	3	10	1	3	10
MTT	28.95	76.73	85.19	34.42	59.81	73.88	33.90	52.92	62.73	13.94	20.28	34.07
DataDAM	40.51	76.60	86.56	34.54	60.37	74.94	35.70	54.56	63.64	16.29	20.51	35.45
Ours	44.70	84.96	87.79	55.80	72.08	80.07	50.20	63.74	68.35	17.29	31.84	43.91

D Additional Ablation Studies

D.1 Point Sorting Strategies

We compared the proposed SADM method against two representative sorting strategies of points. The axis-aligned sorting is simple and heuristic that sorts the coordinates of points, respectively, in the descending order. However, it does not reflect structural or semantic coherence across different samples. The Z-order sorting [3] assigns similar indices to spatially adjacent points. We applied these sorting schemes before the feature extraction. As shown in Table 5, both sorting methods yield marginal improvements over the baseline without sorting, but they often fail to provide consistent semantic alignment across different samples. However, the proposed SADM significantly outperforms them providing reliable matching performance.

Table 5: Performance comparison of different point sorting schemes at PPC 3. For fair comparison, only the primary feature alignment loss \mathcal{L}_α was used, and the rotation optimization was not applied.

Method	ModelNet10	ModelNet40	ShapeNet	ScanObjectNN
Unsorted	75.21	59.96	53.91	20.20
Axis-Aligned	76.44	60.85	53.97	21.80
Z-order [3]	76.55	61.47	55.17	21.76
SADM	83.17	67.52	60.71	27.66

D.2 Rotation Strategies

To further support the effectiveness of the proposed rotation optimization method, we compared widely used rotation augmentation that applies random rotations to the training data to address the rotation variation of 3D point clouds. While this strategy is effective when dealing with datasets showing severe rotational variations, such as ScanObjectNN, it does not achieve good performance on datasets where the 3D objects are fully or partially aligned, such as ModelNet10, ShapeNet, and ModelNet40. As shown in Table 6, the proposed learnable rotation estimation method consistently improves the performance across all datasets, however, the random rotation augmentation introduces unnecessary variation in already aligned datasets, degrading the performance. This highlights the benefit of explicitly modeling the orientation via optimization over the naive augmentation.

Table 6: Performance comparison between the proposed rotation optimization and the random rotation augmentation (Aug.). All experiments were conducted at PPC 3.

Prop.	Aug.	Aligned	Mixed	Rotated
—	—	65.01	59.96	20.42
—	✓	58.92	52.56	31.42
✓	—	74.35	72.08	31.84

E Additional Qualitative Results

Figure 2 shows randomly sampled 3D models from ModelNet40. Figures 3, 4, and 5 visualize the synthesized models at PPC 1 obtained by using DC, DM, and the proposed method, respectively.

While DC and DM maintain the geometric shapes of the original data with only slight movement of some points, the proposed method generates noise-free samples with high visual quality across all classes.

Figure 6 compares the optimization process for DM [12] and the proposed method. DM fails to converge even after multiple training iterations, however the proposed method progressively refines the point cloud models into more structured shapes.

Figure 7 shows the results of the proposed method on ModelNet40 at different PPC values: 1, 3, and 10. As PPC increases, the distilled synthetic datasets exhibit more diverse shapes and orientations. In particular, as shown in the last row, the human class appears in diverse poses. Additional qualitative results of the proposed method on the ShapeNet and ScanObjectNN datasets can be found in Figures 8 and 9, respectively.

References

- [1] George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A Efros, and Jun-Yan Zhu. Dataset distillation by matching training trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.
- [2] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. 2015.
- [3] Guy M Morton. A computer oriented geodetic data base and a new technique in file sequencing. 1966.
- [4] 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/CVF Conference on Computer Vision and Pattern Recognition*, 2017.
- [5] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Proceedings of the Advances in Neural Information Processing Systems*, 2017.
- [6] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017.
- [7] Ahmad Sajedi, Samir Khaki, Ehsan Amjadian, Lucy Z Liu, Yuri A Lawryshyn, and Konstantinos N Plataniotis. Datadam: Efficient dataset distillation with attention matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.
- [8] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.
- [9] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. In *TRANSACTIONS ON GRAPHICS*, 2019.
- [10] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.
- [11] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2015.
- [12] Bo Zhao and Hakan Bilen. Dataset condensation with distribution matching. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023.
- [13] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. In *International Conference on Learning Representations*, 2021.
- [14] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

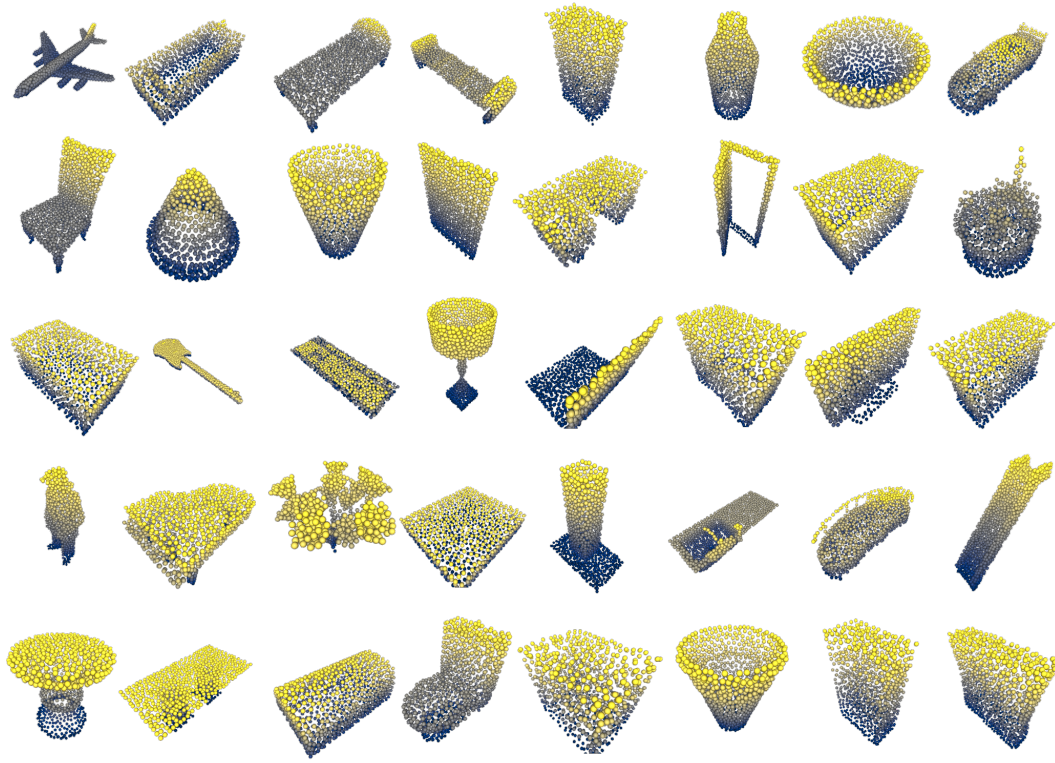


Figure 2: Randomly sampled 3D models of all classes from ModelNet40 [11].

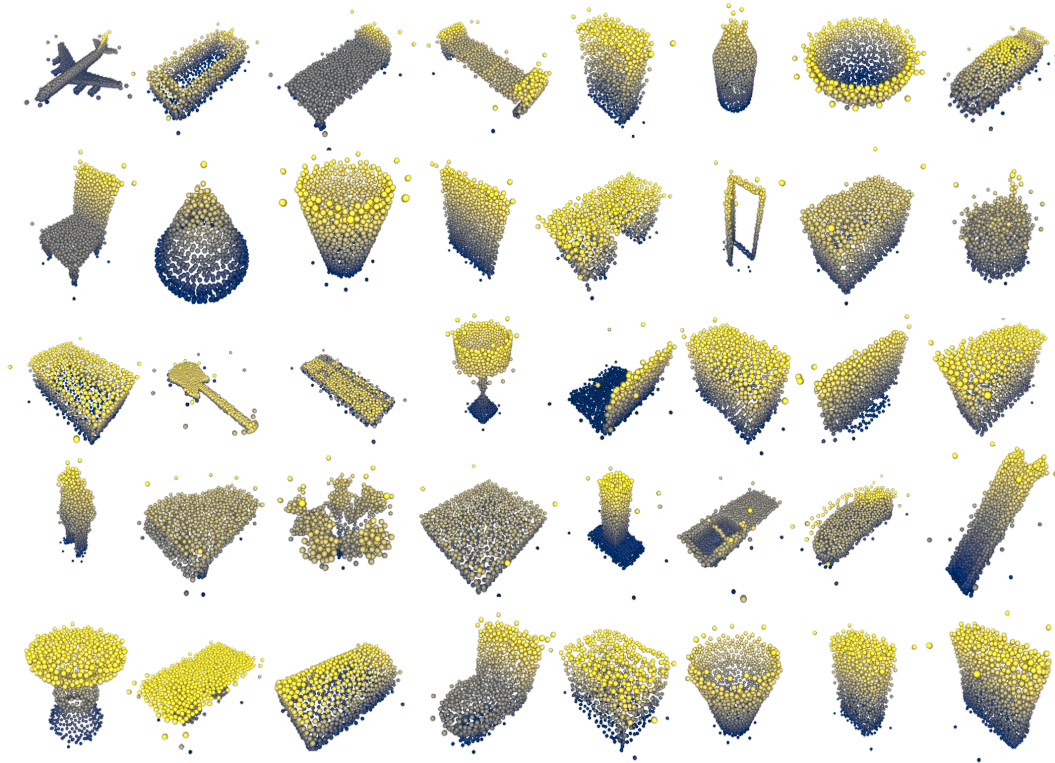


Figure 3: The synthetic dataset distilled from ModelNet40 by using DC at PPC 1.

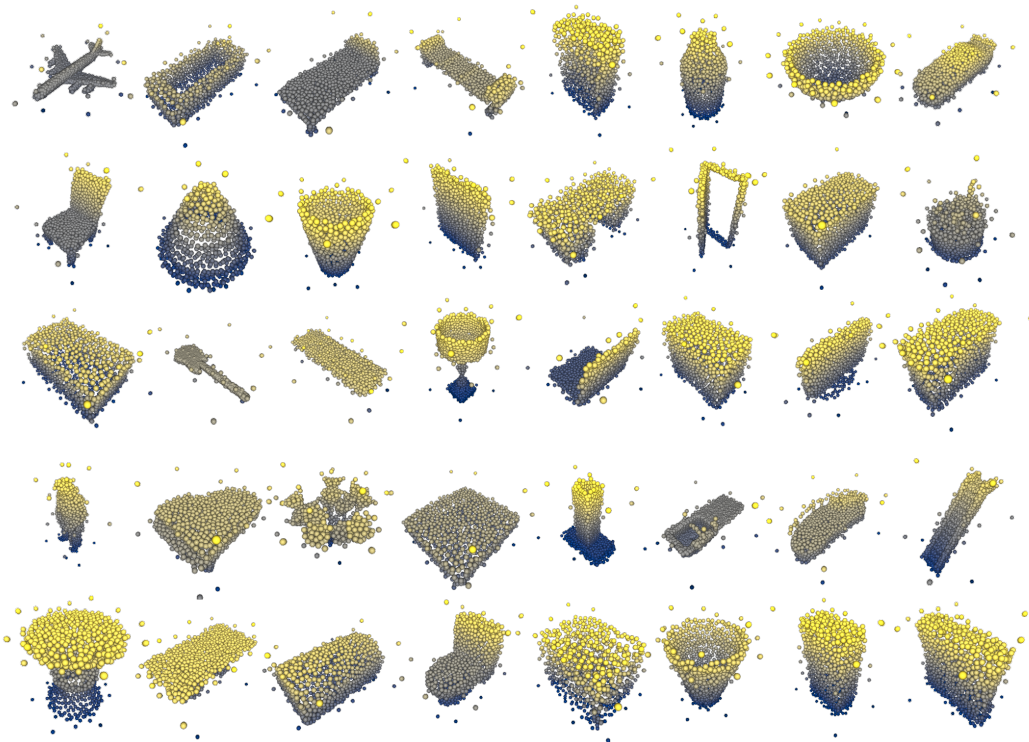


Figure 4: The synthetic dataset distilled from ModelNet40 by using DM at PPC 1.

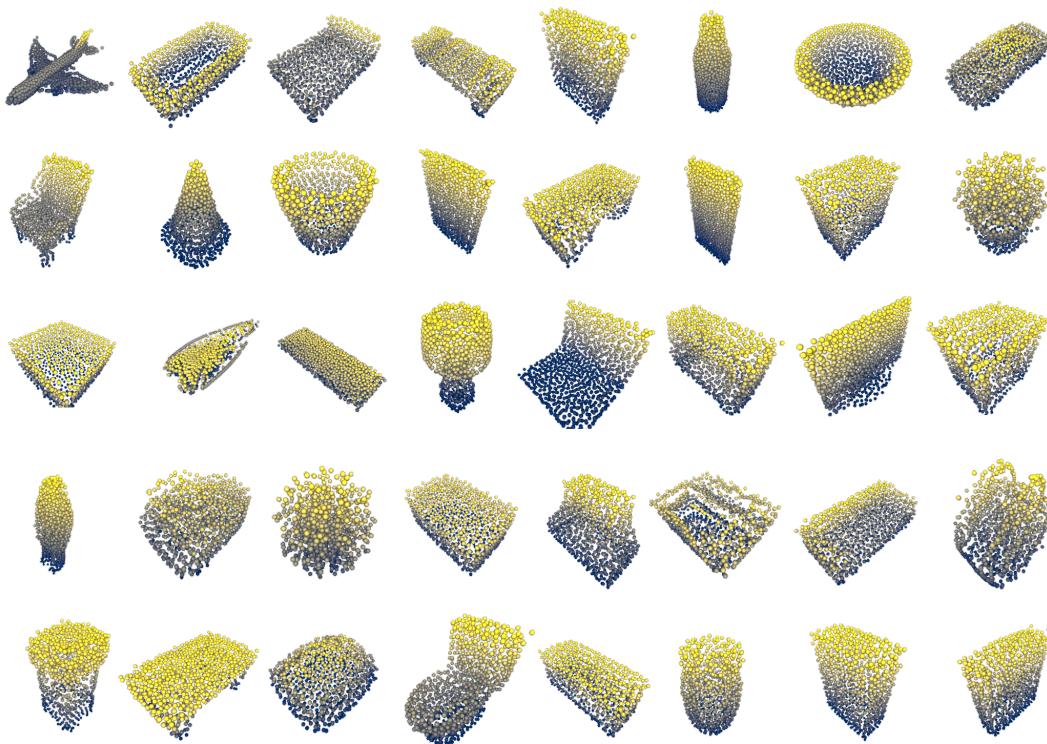


Figure 5: The synthetic dataset distilled from ModelNet40 by using the proposed method at PPC 1.

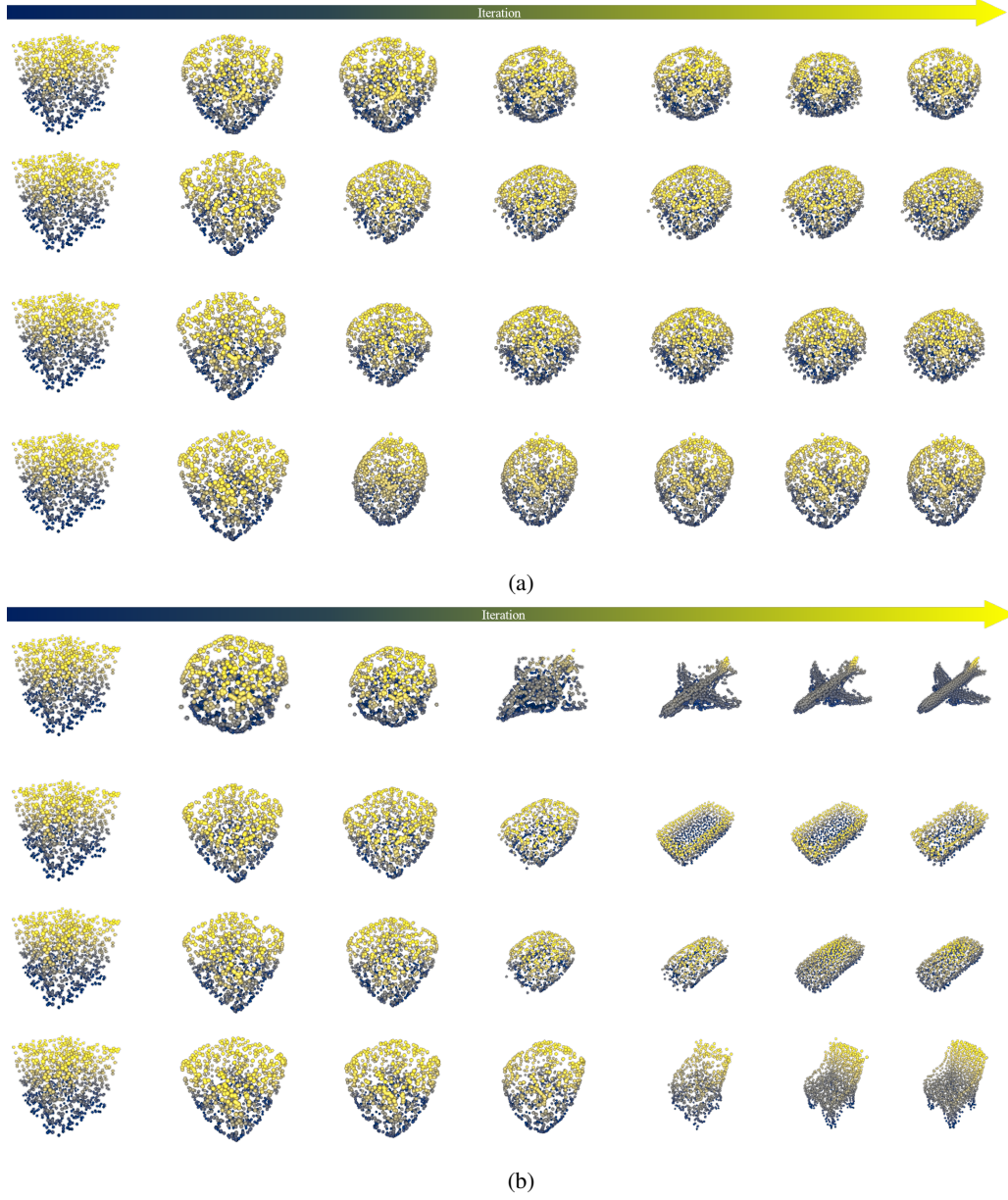


Figure 6: Optimization process in the ModelNet40 at PPC 1 initialized from uniform noise, distilled by using (a) DM [12] and (b) the proposed method.

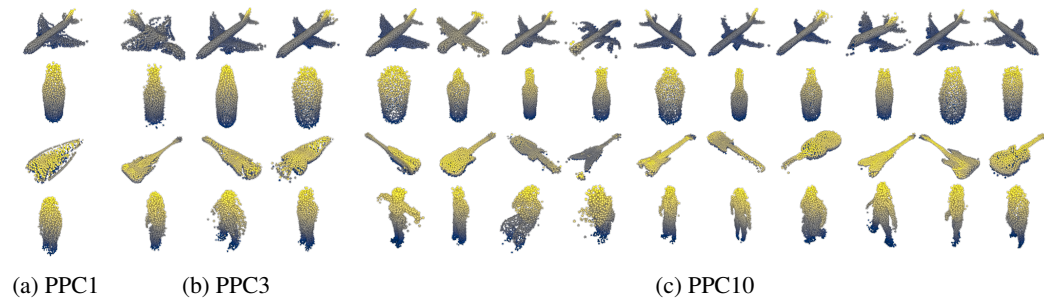


Figure 7: The synthetic dataset distilled from ModelNet40 by using the proposed method at three PPC values of 1, 3, and 10, respectively.

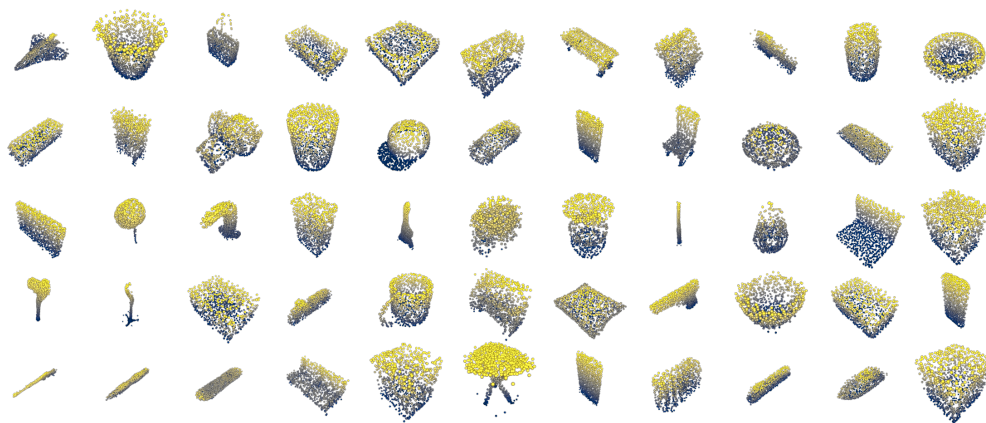


Figure 8: The synthetic dataset distilled from ShapeNet by using the proposed method at PPC 1.

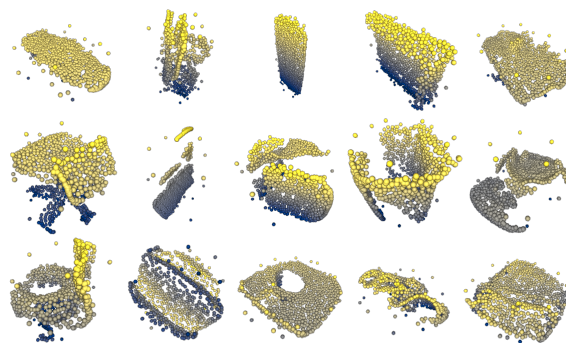


Figure 9: The synthetic dataset distilled from ScanObjectNN by using the proposed method at PPC 1.