ENHANCING POINT CLOUD GENERATION FROM VARIOUS INFORMATION SOURCES BY APPLYING GEOMETRY-AWARE FOLDING OPERATION

by

Yu Lin

APPROVED BY SUPERVISORY COMMITTEE:

Latifur Khan, Chair

Ding-Zhu Du

Nicholas Ruozzi

Vito D'Orazio

To my parents, Mr.Lin and Mrs.Gao

To the moonlight, stars and cosmos

Long may the sun shine

ENHANCING POINT CLOUD GENERATION FROM VARIOUS INFORMATION SOURCES BY APPLYING GEOMETRY-AWARE FOLDING OPERATION

by

YU LIN, BS

DISSERTATION

Presented to the Faculty of

The University of Texas at Dallas

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY IN

COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT DALLAS

May 2022

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Latifur Khan, for the great support, guidance and patience during my PhD career. He continuously provided insightful suggestions and is willing and enthusiastic to assist in any way he could throughout the research projects.

I would like to extend my gratitude to my dissertation committee members, Dr. Vito D'Orazio, Dr. Nicholas Ruozzi, and Dr. Ding-Zhu Du, for their effective guidance and valuable comments on my papers.

I also need to thank my fellow lab mates, Bo, Hemeng, Jinghui, Yang, Yibo, Yifan, Yigong, Zhuoyi, and my oversea friends, Bonian, Wenxiong, Xiaoqiang, Xiaoyi for their great assistance in my life and research work.

To my family, thank you for encouraging me in all of my pursuits. In particular, I would like to express my deepest love and thanks to my father, Mr. Lin, and my mother, Mrs. Gao, I could not finish this journey without your support and encouragement. Thank you.

Finally, the research reported herein was supported in part by NSF awards DMS-1737978, DGE-2039542, OAC-1828467, OAC-1931541, DGE-1906630; and an IBM faculty award (Research).

March 2022

ENHANCING POINT CLOUD GENERATION FROM VARIOUS INFORMATION SOURCES BY APPLYING GEOMETRY-AWARE FOLDING OPERATION

Yu Lin, PhD
The University of Texas at Dallas, 2022

Supervising Professor: Latifur Khan, Chair

A plethora of cutting-edge computer vision and graphic applications, such as Augmented Reality (AR), Virtual Reality (VR), automatic vehicles, and robotics, require rapid creation and access to abundant 3D data. Among various 3D data representations, e.g., RGB images, depth images, or voxel grids, point cloud attracts considerable attention from the research community because it offers additional geometric, shape, and scale information in comparison with 2D images and demands less computational resource to process in contrast to other 3D representations, e.g., voxel grids, octree, or triangle meshes. Unfortunately, even with the increasing availability of 3D sensors, the size and variety of 3D point clouds datasets pale when compared to the vast size datasets of other representations. Therefore, it will benefit many applications if we can generate point clouds from other information sources.

Point cloud generation is a sub-field of 3D reconstruction, which aims to generate a complete 3D object from other information sources. Conventional methods generally focus on 2D images and heavily rely on the knowledge of multi-view geometry, while multiple 2D views of a target 3D object usually are inaccessible in many real-world scenarios. On the contrary, recent deep learning approaches either dedicate to 3D representations with regular structures, such as voxel grids and octrees, and thus suffer from resolution and scalability issues, or unconsciously ignore the crucial 3D prior knowledge and lead to sub-optimal solutions.

To address the aforementioned drawbacks, we explore the possibilities to improve the point cloud generation by developing advanced folding operations and geometry-aware (3D-prior-aware) reconstruction networks in this dissertation. Specifically, we start with a novel point cloud generation framework TDPNet that reconstructs complete point clouds by employing a hierarchical manifold decoder and a collection of latent 3D prototypes. Later, we find that applying vanilla folding operation is insufficient for a realistic reconstruction, and using KMeans centroids as the prototype features is unstable and lacks interpretability. Inspired by these observations, we further introduce a novel framework equipped with a collection of Learnable Shape Primitives (L-SHAP), which encode the crucial 3D prior knowledge from training data through an additional folding operation. On the other hand, it's beneficial to many applications if point clouds can be generated in a few-shot scenario. We tackle this problem by a novel few-shot generation framework FSPG, which simultaneously considers class-agnostic and class-specific 3D priors during the generation process. Finally, we observe that conventional folding operations are implemented by a simple shared-MLP, which increases training difficulty and limits the network's modeling capability. In order to solve this problem, we incorporate the popular Transformer architecture into a novel attentional folding decoder AttnFold and introduce a Local Semantic Consistency (LSC) regularizer to further boost the model's capability.

Based on our research, we demonstrate that learning flexible data-driven 3D priors and adopting advanced folding operations are effective for point cloud generation under different problem settings.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION [1] [2]

A plethora of cutting-edge computer vision and graphic applications, such as Augmented Reality (AR) (de Souza Cardoso et al., 2020), Virtual Reality (VR) (Stets et al., 2017; Blanc et al., 2020), automatic vehicles (Yue et al., 2018; Cui et al., 2021), and robotics (Li et al., 2020), require rapid creation and access to abundant 3D data. These 3D data can be described by various data representations where the structure and the geometric properties vary from one representation to another. In this dissertation, we split 3D data into two main categories based on the dimension of their structure: 2D representations and 3D representations. To be more specific, "2D representations" describe the 3D data via a set of 2D descriptors (Wang et al., 2004; Jin et al., 2005; Lai et al., 2011; Su et al., 2015), e.g., RGB images, Depth images, and Multi-view images, yet "3D representations" model the 3D data in the 3D Euclidean space, e.g., voxel grids (Wang et al., 2019), polygon meshes (Masci et al., 2015), and point clouds (Qi et al., 2017). It's easy to observe that 3D representations generally provide richer geometric, shape, and scale information, thus would benefit several applications in the real-world scenario.

Among various 3D representations, point cloud is becoming more and more popular because of its expressiveness, compactness and homogeneousness (Ahmed et al., 2018). As shown in Figure 1.1, point cloud is able to seamlessly describe 3D object with a much higher resolution in comparison to voxel grids and octrees. On the other hand, point cloud can be considered as an intermediate representation of polygon mesh because it provides

[1] This chapter contains material previously published as: Yu Lin, Yigong Wang, Yifan Li, Zhuoyi Wang, Yang Gao, and Latifur Khan. "Single View Point Cloud Generation via Unified 3D Prototype". In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 3, pp. 2064-2072. 2021

[2] This chapter contains material previously published as: Yu Lin, Jinghui Guo, Yang Gao, Yifan Li, Zhuoyi Wang, and Latifur Khan. "Generating Point Cloud from Single Image in The Few Shot Scenario". In *Proceedings of the 29th ACM International Conference on Multimedia*, pp. 2834-2842. 2021

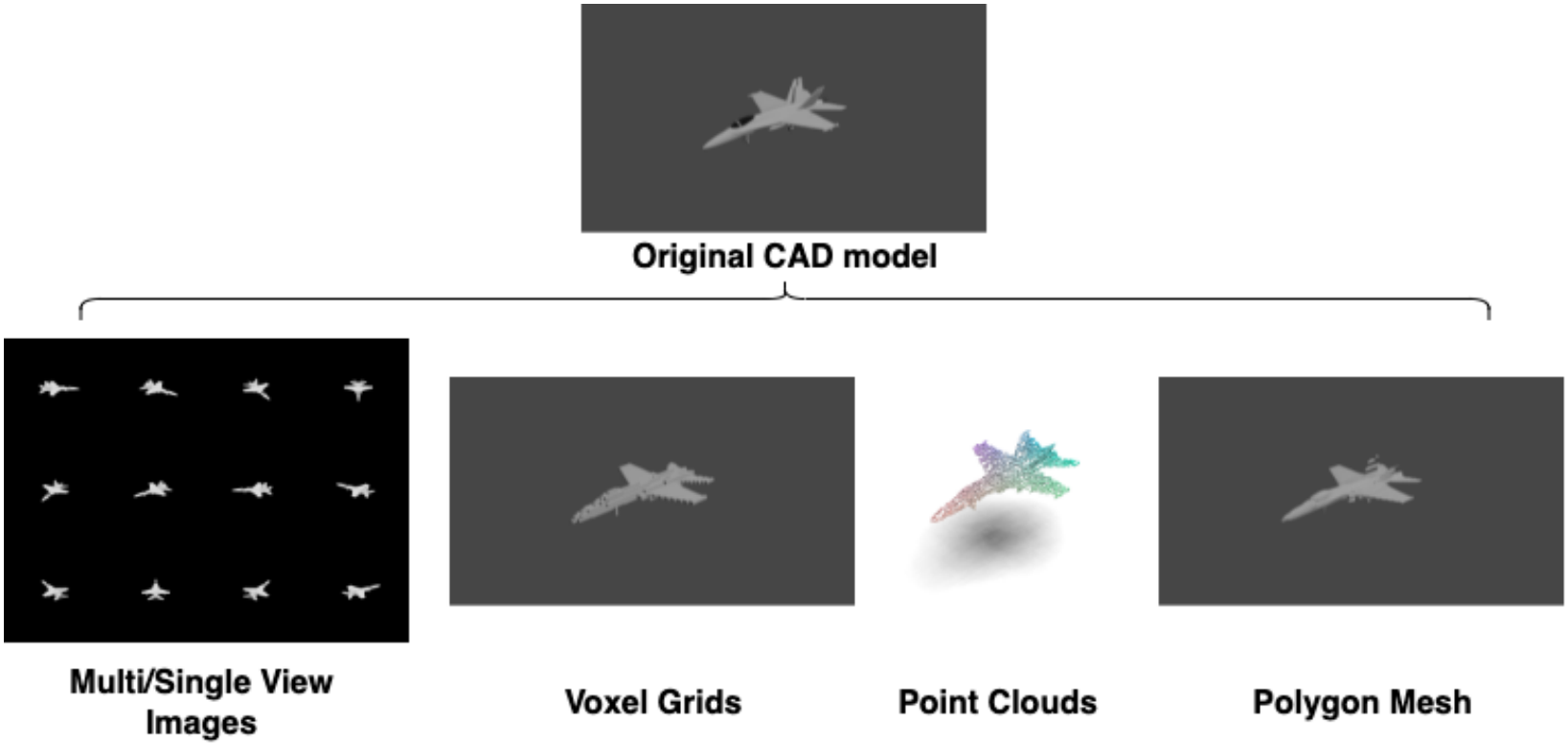

Figure 1.1. An example of 3D data and its different 2D and 3D representations. This CAD model is sampled from ShapeNet (Chang et al., 2015) dataset.

similar information, and more importantly, demands less computational resource (Lin et al., 2021). Considering these advantages, point cloud is extremely suitable for many time sensitive applications. For instance, self-driving may benefit from a complete point cloud that describes the surrounding environment because such a point cloud provides more information and can be processed in a short amount of time. Nevertheless, 3D sensors that directly produce point clouds, such as LiDARs, are still much more extravagant comparing to the traditional cameras. Generating point cloud from other data representations, 2D images in particular, therefore, is a demanding yet promising research direction.

Towards this goal, this dissertation addresses the challenges in point cloud generation from several aspects. We start with a single-view point cloud generation problem by considering explicit 3D shape priors. The first two works adopt a folding-based neural network as the point cloud generator and encode crucial 3D priors information, in a hard-coded and data-driven way, respectively. In the third work, we utilize the aforementioned techniques to simultaneously model the class-specific and class-agnostic 3D shape priors and solve a challenging few-shot generation problem. Finally, we improve the folding-based point cloud generator by introducing a local semantic consistency regularizer and employing a Transformer-like (Vaswani et al., 2017) attention mechanism, which allows the network to aware the manifolds' global context and generate a more realistic point cloud.

1.1 3D Reconstruction

3D reconstruction aims to generate complete 3D data (profile) that describe the shape and appearance of objects, either real or artificial, from various information sources. For example, stereo vision takes multiple images as the information source and obtains the geometric information of the target object by following the vision mechanism of human (Cardenas-Garcia et al., 1995). Before the deep learning era, traditional 3D reconstruction methods mathematically reverse the 2D-3D projection process and heavily rely on the availability

of multiple images, from either one (Zhang et al., 1999; Cheung et al., 2003; Lobay and Forsyth, 2006) or multiple (Esteban and Schmitt, 2002; Geiger et al., 2011; Schmid et al., 2012) viewpoints. Even though these methods have achieved promising performance, they are limited by the coverage of images.

Inspired by the success of deep learning techniques in the image domain, several state-of-the-art frameworks leverage deep neural networks to learn a 3D shape from images (Choy et al., 2016; Rezende et al., 2016; Gadelha et al., 2017; Tulsiani et al., 2017). These methods avoid the complex stereo correspondence and camera calibration by solving the problem from a different perspective. Specifically, they reformulate the 3D reconstruction problem into a recognition problem and learn the 3D prior knowledge through neural networks (Abedin et al., 2006; Lavee et al., 2007; Han et al., 2019). In conclusion, the exciting and promising results provided by these methods demonstrate the effectiveness of deep learning on 3D reconstruction.

1.1.1 Point Cloud Reconstruction and Generation

As a sub-field of 3D reconstruction, point cloud generation chooses point cloud as the output representation of 3D shapes. Literally, point cloud describe 3D objects through a collection of points, each denotes a 3D coordinate and optional attributes, such as normal, color, and texture. Compared to voxel grids and polygon meshes that require exceeding computational resources, point clouds can be processed through a relatively simple network. Nevertheless, it's not trivial to extend deep learning techniques to the point cloud domain because of their irregular structure.

On the other hand, point clouds are practically captured from the object's surface, therefore, any surface reconstruction framework (Gotsman et al., 2003; Praun and Hoppe, 2003; Sheffer et al., 2007; Monti et al., 2017; Wang et al., 2018; Pontes et al., 2018; Groueix et al., 2018) applies to this task. To be more specific, we can first reconstruct the object's

surface through aforementioned methods and conduct random sampling to generate a point cloud. Such a pipeline, however, inevitably requires more computational resources due to the connectivity of the surface and thus is less efficient compared to the point-based methods (Fan et al., 2017; Jiang et al., 2018; Gadelha et al., 2018; Lin et al., 2018). Even great progress has been made, the aforementioned methods do not explicitly consider the 3D shape prior and assume the neural network will learn the information implicitly. Conversely, employing explicit and flexible 3D shape prior and designing the network architecture carefully would lead to a better reconstruction (Lin et al., 2021).

1.1.2 Point Cloud Generation with Explicit 3D Prior

Predicting a complete 3D shape from other information sources, especially images, is a long-standing conundrum in the computer vision community. This problem is ill-posed, and prior knowledge is mandatory because an RGB image (or multi-view images, partial point cloud) contains deficient information for a complex 3D model (Fan et al., 2017; Kato and Harada, 2019). As we discussed in the previous section, state-of-the-art deep learning frameworks generally assume the neural network will learn the shape prior information implicitly and treat other information sources and 3D shapes equally. For example, representative image-to-point-cloud frameworks (Fan et al., 2017; Groueix et al., 2018; Yang et al., 2018) adopt a classical auto-encoder architecture, which computes the latent representation of an input image and decodes it into a point cloud by using this latent representation only.

To address these limitations, we first propose a framework named Three Dimensional Prototype Network (TDPNet) (Lin et al., 2021). This method utilizes a set of latent 3D prototype features obtained from an external point cloud dataset, which explicitly encodes the 3D shape prior information. Moreover, we introduce a hierarchical manifold decoder that encourages diverse reconstruction and avoids mode collapse by decoding each prototype separately. Although the proposed method achieves promising results, we observed that

defining the 3D prototype features with KMeans centroids of external point cloud dataset is unstable and lacks interpretability. Therefore, we further introduce an advanced point cloud generation framework with Learnable Shape Primitives (L-SHAP), which explicitly encodes the 3D shape prior information from training data through an additional folding operation. Our experiments show that both proposed methods can effectively obtain superior performance compared to the existing methods, on the famous ModelNet and ShapeNet datasets.

1.1.3 Few-shot Point Cloud Generation

In many real-world scenarios, there exist copious 2D images with no corresponding 3D point cloud. As a consequence, it would be extremely valuable if we are able to generate point clouds merely from a single RGB image. Following the idea of explicit 3D shape priors, we argue that point clouds can be reconstructed from a single image with a class-specific prototype feature obtained from limited support samples (point clouds) and class-agnostic shape primitives learned from training data. However, previous state-of-the-art few-shot generation methods (Wallace and Hariharan, 2019; Michalkiewicz et al., 2020) focus on the regular voxel representation and can not be extended to the point cloud domain easily. To this end, we propose a novel few-shot single-view point cloud generation framework accompanied with a novel episodic training strategy. The results of experiments on ModelNet and ShapeNet demonstrated that our method outperforms state-of-the-art approaches by a large margin.

1.1.4 Advanced Folding Operation

Recent deep point cloud generation approaches generally follow the architecture of auto-encoder. Considerable representation learning frameworks on various domains haven been proposed, yet the design of decoder also plays a crucial role for a realistic point cloud generation. Pioneers (Fan et al., 2017; Achlioptas et al., 2018) in this direction adopt simple

fully connected layers to generate a coarse point clouds, which inevitably suffer from the scalability issue due to the magnitude of network parameters and convergence speed. More recently, several folding-based methods (Groueix et al., 2018; Yang et al., 2018; Lin et al., 2021) embrace the idea of manifold deformation and deform one or multiple canonical 2D grids onto the target surface. It's worth noting that these methods approximate the deformation function through a shared MLP, thus ignored the spatial interactions between each point during the generation process, leading to undesired over-complicated components. To address this issue, we propose an attentional point cloud decoder that applies the self-attention mechanism to aggregate latent features and global context of source surfaces. The evaluation results on both PointDA and ShapeNet datasets show the significantly improved effectiveness compared with baselines.

1.2 Contribution of this dissertation

In summary, the contribution of this dissertation is as follows:

1.2.1 Point Cloud Generation with 3D Prototypes

- We propose a deep learning framework, TDPNet, to solve the image-to-point-cloud generation problem. It compensates the missing information of images by combining 2D image features and 3D prototype features in the hidden space.

- The proposed TDPNet adopts a unified 3D prototype schema to efficiently utilize the rich structural 3D information.

- We conduct extensive experiments to verify the effectiveness of our method, both quantitatively and qualitatively.

1.2.2 Point Cloud Generation with Learnable Shape Primitives

- To the best of our knowledge, we are the first that bring learnable shape primitives (3D priors), L-SHAP, into the single-view point cloud reconstruction scenario.
- Based on the task requirement (e.g., dense point cloud generation) and the initial point set's topology, we introduce two alternative shape primitives: continuous shape primitives and discrete shape primitives.
- We empirically demonstrate the superiority of L-SHAP over existing state-of-the-art solutions on two typical benchmarks including Modelnet and ShapeNet.

1.2.3 Few-shot Point Cloud Generation

- To the best of our knowledge, FSPG is the first deep learning framework that tackles the challenging few-shot single-view reconstruction problem in the point cloud domain.
- We introduce a novel network architecture that learns class-specific and class-agnostic shape priors simultaneously, and train the network with an advanced episodic training strategy.
- The proposed episodic training strategy successfully avoids the expensive finetuning process on unseen novel classes, therefore significantly boost the model's efficiency.

1.2.4 Advanced Folding Operation

- We are the first to present an attentional folding-based point cloud generation framework, which considers the spatial interaction between points sampled from the source surfaces.
- In addition to the traditional point-wise loss function, we introduce a novel semantic consistency regularizer to further improve the generation performance.

1.3 Outline of this dissertation

The rest of the dissertation is organized as follows. Chapter 2 provides the background of baseline approaches and evaluation protocol in this dissertation. Chapter 3 discusses the framework that adopts 3D prototype features from existing point cloud datasets. Chapter 4 presents our solution for integrating flexible 3D shape prior knowledge into point cloud generation. Chapter 5 describes the few-shot point cloud generation framework. Chapter 6 introduces a novel attentional folding-based operation. Chapter 7 summarizes this dissertation and discusses the future work.

CHAPTER 2

BACKGROUND [1] [2]

In this chapter, we present relevant background information of existing and previous point cloud learning/generation methods.

2.1 Point Cloud Representation Learning

Mathematically, a point cloud S can be formulated as a set of points $S = \{p_i\}_{i=1}^{N}$, where N denotes its cardinality and $p_i = (x_i, y_i, z_i, a_i^{(1)}, a_i^{(2)}, \ldots, a_i^{(k)})$ is a point in the 3D Euclidean space with coordinate (x_i, y_i, z_i) and k optional attributes $(a_i^{(1)}, a_i^{(2)}, \ldots, a_i^{(k)})$ (e.g., color and normal). Representation learning is crucial to understanding and utilizing point clouds in deep learning since most deep learning techniques cannot consume "set" objects directly. For instance, a 2D image is usually denoted by a $C \times H \times W$ pixel matrix, where C is the number of channels (e.g., C=3 for RGB images), H and W are the height and width of the image. Such well-ordered data structures are amenable to many famous deep learning methods, such as Convolutional Neural Network (CNN) (Breen et al., 2002; LeCun et al., 2015), and lead to promising results. On the other hand, these famous techniques may not be applicable in the point cloud domain due to the permutation invariant property of set, which means that the order of elements should not affect the output. Let $f(\cdot)$ be a trained deep learning model, $X = \{x_1, \ldots, x_n\}$ be the input set, and π be a permutation. Permutation invariant property can be formally described by:

---

[1] This chapter contains material previously published as: Yu Lin, Yigong Wang, Yifan Li, Zhuoyi Wang, Yang Gao, and Latifur Khan. "Single View Point Cloud Generation via Unified 3D Prototype". In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 3, pp. 2064-2072. 2021

[2] This chapter contains material previously published as: Yu Lin, Jinghui Guo, Yang Gao, Yifan Li, Zhuoyi Wang, and Latifur Khan. "Generating Point Cloud from Single Image in The Few Shot Scenario". In *Proceedings of the 29th ACM International Conference on Multimedia*, pp. 2834-2842. 2021

$$\forall \pi, f(\{x_1, \ldots, x_n\}) = f(\{x_{\pi(1)}, \ldots, x_{\pi(n)}\}) \tag{2.1}$$

Several state-of-the-art frameworks are proposed to tackle this problem on point cloud domain by following the DeepSet (Zaheer et al., 2017) paradigm, which theoretically demonstrates the requirements of permutation invariant network. Specifically, a set function, $f(\cdot)$, learnt by a neural network, is invariant to the permutation of elements in the input set X, iff it can be decomposed in the form $\rho(\sum_{x \in X} \phi(x))$, for suitable transformations ρ and ϕ.

Point analysis and *neighborhood analysis* are the two main categories in the deep point cloud representation learning. The former schema treats each point independently and aggregates the information of all points with a pooling function, whereas the second schema considers the ambient signal of each point by locally applying convolution operations. We will then briefly describe some representative approaches of each category.

2.1.1 Point Analysis Methods

Point analysis methods generally compute the feature of each point separately and aggregate their information by applying some permutation invariant pooling function, e.g., $\max(\cdot)$. The feature extractor part is shared by all points, thus satisfies the permutation invariant requirement and lead to efficient and effective representation computation.

PointNet (Qi et al., 2017)

PointNet is the first point analysis approach, which learns the shape descriptors of point cloud via a collection of shared nonlinear functions (implemented by a simple 3-Layer MLP) and a symmetric aggregation function. Instead of sorting the input points into a canonical order or training a RNN with all possible permutation sequences, PointNet inspiringly adopts a simple symmetric function to efficiently aggregate the information from each point.

Specifically, let $S \in \mathbb{R}^{N\times 3}$ be an input point cloud, PointNet first applies an input transformation (implemented by a mini-PointNet that outputs a 3×3 affine transformation matrix) and projects it into a feature space, $S_{feat} \in \mathbb{R}^{N\times 64}$. A feature transformation (implemented by a mini-PointNet that outputs a 64×64 affine transformation matrix) follows and the transformed feature matrix is further projected into another feature space, $S_{feat'} \in \mathbb{R}^{N\times 1024}$. Finally, a max pool layer is applied to get the global feature of input point cloud, $S_g \in R^{1024}$.

PointNet++ (Qi et al., 2017)

PointNet++ is an advanced version of PointNet that hierarchically extracts point features by sampling the point cloud in different granularities. To be concrete, PointNet++ is a sequence of *set abstraction* components, each has a sampling layer, a grouping layer, and a small PointNet layer. The sampling layer conducts Farthest Point Sampling (FPS) to generate a higher-granularity point set, e.g., applying FPS on a point set $S \in \mathbb{R}^{N\times d}$ results in $S' \in \mathbb{R}^{M\times d}$, where d is the feature dimension of each point and $M < N$. Later, K Nearest Neighborhood (KNN) is applied on the original point set S by using S' as the centriods, leading to a $N \times K \times d$ data matrix. This data matrix is handled by the followed PointNet layer, which extracts the local information of each centroid by applying a shared PointNet among all M centroids. Output data size is now $M \times d'$, where d' is the new feature dimension.

Therefore, the shape information of a point cloud in different granularities (e.g., from coarse to smooth) is captured by different *set abstraction* block. By repeating this *set abstraction* process multiple times and applying a max pool operation, the global feature of input point cloud can be finally obtained, $S_g \in R^{1024}$.

2.1.2 Neighborhood Analysis Methods

Although aforementioned *point analysis* methods achieve amazing results, they generally ignore the ambient signal of each point and are surpassed by the *neighborhood analysis methods*.

ECC (Simonovsky and Komodakis, 2017)

This Edge-Conditioned Convolutional neural network (ECC) considers the input point cloud S as a graph $G = (V, E)$, where V denotes all points $p \in S$ and E is a set of directed edges built by connecting each vertex (point) i to all vertices j in its spatial neighborhood (e.g., ball query or KNN). The label for each edge is represented by a 6D label vector:

$$L(j,i) = (\delta_x, \delta_y, \delta_z, \|\delta\|, \arccos \frac{\delta_z}{\|\delta\|}, \arctan \frac{\delta_y}{\delta_x}) \tag{2.2}$$

where $\delta = p_j - p_i$. Let $l \in \{0, \ldots, l_{max}\}$ be the layer index in a graph neural network. $X^l : V \to R^{d_l}$ assigns labels to each vertex and $L : E \to R^s$ assigns labels to each edge. X^0 simply is the original input point cloud G. Recall that the coordinate information is captured by the label of edge, the label of point p_i, $X^0(i)$, would be its optional attributes, $(a_i^{(1)}, a_i^{(2)}, \ldots, a_i^{(k)})$, or 0 if there is no available attribute. The filtered signal of a vertex i at layer l is defined as

$$X^l(i) = \frac{1}{\|N(i)\|} \sum_{j \in N(i)} F^l(L(j,i); w^l) X^{l-1}(j) + b^l \tag{2.3}$$

where $N(i)$ denotes the neighbors of vertex i and $F^l(L(j,i); w_l)$ computes the weight factor of $X^{l-1}(j)$ by considering the information of edge (i,j). The final point cloud representation is obtained by applying a max pooling operation after layer l_{max}. Such network architecture allows the model to effectively consider the local information of each point, leading to a better representation.

Pointwise CNN (Hua et al., 2018)

Instead of creating a large graph for the whole point cloud, this framework directly applies the convolution operation on the input point clouds. A ball query is first conducted on each point to find its neighborhood and sort them into different quadrants. For a specific network layer l, the activation of point i can be formulated as:

$$X^l(i) = \sum_k w_k \frac{1}{\|\Omega_i(k)\|} \sum_{j \in \Omega_i(k)} X^{l-1}(j) \tag{2.4}$$

where k iterates over all sub-domains (quadrants) in the kernel support. $\Omega_i(k)$ denotes the k-th quadrant of kernel centered at point i and j is a point in that quadrant. Moreover, the max pool layer is no longer required since the point cloud is first sorted into a canonical order.

DGCNN (Wang et al., 2019)

More recently, this Dynamic Graph CNN (DGCNN) is proposed to efficiently exploit local geometric structures of point cloud by constructing a local neighborhood graph and applying a convolution operation named as *EdgeConv*. Not like ECC (Simonovsky and Komodakis, 2017), such framework avoids the artificial label of edges and learns the edge information through a trainable function.

Specifically, this method first constructs the KNN graph, including self-loop, for each point at each network layer. For a given network layer l, the edge feature between two vertices (points), i and j, is defined as $E^l_{(i,j)} = h^l_\Theta(X^l(i), X^l(j))$ and $h^l_\Theta(\cdot,\cdot)$ is a trainable function. The *EdgeConv* operation is a channel-wise symmetric aggregation operation Υ, e.g., summation or maximization, and the output of EdgeConv at i-th vertex is $X(i) = \Upsilon_{j \in N(i)} h_\Theta(X(i), X(j))$, where $N(i)$ denotes set of points that connect to point i. In conclusion, DGCNN computes the global representation of a given point cloud by sending the input to several EdgeConv blocks, concatenating the output of each block, and passing them to a max pool layer.

2.2 Deep Point Cloud Generation

In opposite to point cloud representation learning that is trying to learn a hidden representation of a given point cloud, researchers on point cloud generation area are focusing on re-generate a complete point cloud from other information resources (most of them can be considered as a latent vector). Here we list some representative approaches.

PSGN (Fan et al., 2017)

This Point Set Generation Network (PSGN) combines an MoN loss and a powerful two-branch decoder architecture to restore point clouds from one image. Specifically, a CNN is adopted as the image feature extractor to compute the latent representation of a given image. The resulting image representation is then consumed by the MLP-branch of the decoder to generate point set S_1 and the intermediate CNN activation values are consumed by the Deconv-branch to generate another point set S_2. The final output point cloud is just the union of S_1 and S_2.

Moreover, to address the ambiguous 3D shape caused by the information loss of 2D images, the Min-of-N (MoN) loss is introduced to better model such uncertainty. For a given image I, the point cloud decoder $G(\cdot)$ generate n predictions by perturbing the input pixels with n random vectors r_j. The MoN loss is formally defined as:

$$L_{MoN} = \min_{r_j \sim N(0,1), 1 \leq j \leq n} \{d(G(I, r_j), S^{gt})\} \tag{2.5}$$

where $d(\cdot, \cdot)$ is a predefined pointwise distance metric and S^{gt} means the ground truth point cloud. The intuition behind this loss function is that the minimum of the n distances between each prediction and the ground truth must be small.

FoldingNet (Yang et al., 2018)

In opposite to PSGN, which directly maps a latent vector to a point cloud, FoldingNet tackles this problem from a different perspective. It deforms a canonical 2D grid onto the underlying 3D object surface of a point cloud and the deform function is approximable by MLPs.

After obtaining the latent representation, $e \in \mathbb{R}^d$, of an information source, e.g., point cloud or RGB image, by using a pretrained feature extractor, FoldingNet replicates this representation N times and concatenates them with N point coordinates sampled from an unit square, $[0, 1]^2$. The output data now has size $N \times (d + 2)$ and will be consumed by a shared-MLP to generate a $N \times 3$ point cloud. The intuition behind this operation is that any 3D object surface could be transformed into a 2D plane, and the inverse procedure will map a 2D plane back to the 3D object surface. Moreover, FoldingNet actually applies this operation twice to further boost the framework's modeling capability.

AtlasNet (Groueix et al., 2018)

Without loss of generality, AtlasNet shares the same idea of FoldingNet: generating a point cloud can be considered as generating a surface of a 3D shape.

However, AtlasNet can be considered as an advanced version of FoldingNet since FoldingNet restricts itself to one manifold (2D unit square) and AtlasNet deforms multiple 2D grids with MLPs. Nevertheless, AtlasNet only conducts the folding operation once rather than twice in FoldingNet, which results in limited shape modeling capability if utilizing only one 2D unit square.

2.3 Few Shot Learning in Point Cloud

Since we are going to talk about the application of point cloud generation in the few-shot configuration, we further provide some background of few-shot learning. Existing Few-Shot

Learning (**FSL**) aims to endow a model with strong adaptation ability through episodic training (Vinyals et al., 2016; Snell et al., 2017; Wang et al., 2020), and produce a generalizable result on novel classes with only a few labeled samples. FSL (Wang et al., 2020; Sharma and Kaul, 2020) can be roughly divided into three perspectives: *metric-based*, *model-based* and *optimization-based*. This dissertation focuses on the *metric-based* approaches which learn the metric space by computing Euclidean distances to prototypical representations of each class.

A class-specific prototypical representation is ideal to encode the 3D shape prior of that class, this idea has been explored by two representative few-shot 3D reconstruction approaches. More specifically, Wallace *et al.* (Wallace and Hariharan, 2019) propose to generate target voxel grids by refining a template voxel computed from support set, which is considered as a class-specific shape prior, with 2D image features. CGCE (Michalkiewicz et al., 2020) invent a hierarchical shape prior model, where each class-specific shape prior is built from a set of learnable parameters. The trained network will be finetuned on the novel classes to obtain corresponding class-specific shape prior and generate point cloud from it. These two methods, however, are limited to the voxel grid domain and consider only the class-specific shape prior. To overcome these limitations, we propose a powerful architecture that considering both class-specific and class-agnostic shape prior for the purpose of few-shot point cloud generation.

CHAPTER 3

POINT CLOUD GENERATION VIA UNIFIED 3D PROTOTYPE [1]

3.1 Approach

In this chapter, we assume the information source of target point cloud is one of its 2D views (RGB images) and focus on designing a novel architecture that allows efficient image-to-point-cloud generation and closes the gap between 2D and 3D features. Specifically, our solution is a unified point cloud generation framework based on 3D prototypes, named as **T**hree **D**imensional **P**rototype **Net**work (TDPNet).

Figure 3.1 demonstrates the intuition of our method. Instead of generating point clouds merely from 2D images, our method aims to generate point clouds by combining 2D image features and class-specific prototypes that encode 3D prior knowledge. Noted that most of the existing approaches generally follow the autoencoder (AE) architecture and replace the encoder part with an image encoder. Nevertheless, a 2D image, as the projection of corresponding 3D shape, naturally contains limited information comparing to a complete point cloud. Inspired by the observation that many shapes from a specific category generally have similar 3D structures, e.g., airplane wings and cabins, we suggest to compensate the information loss by adopting corresponding class-specific prototypes.

3.1.1 Framework Overview

Problem Setting

The objective of our framework is to reconstruct a complete point cloud from a single 2D projection, with the aids from existing point cloud datasets. A point cloud is presented as

[1] This chapter contains material previously published as: Yu Lin, Yigong Wang, Yifan Li, Zhuoyi Wang, Yang Gao, and Latifur Khan. "Single View Point Cloud Generation via Unified 3D Prototype". In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 3, pp. 2064-2072. 2021

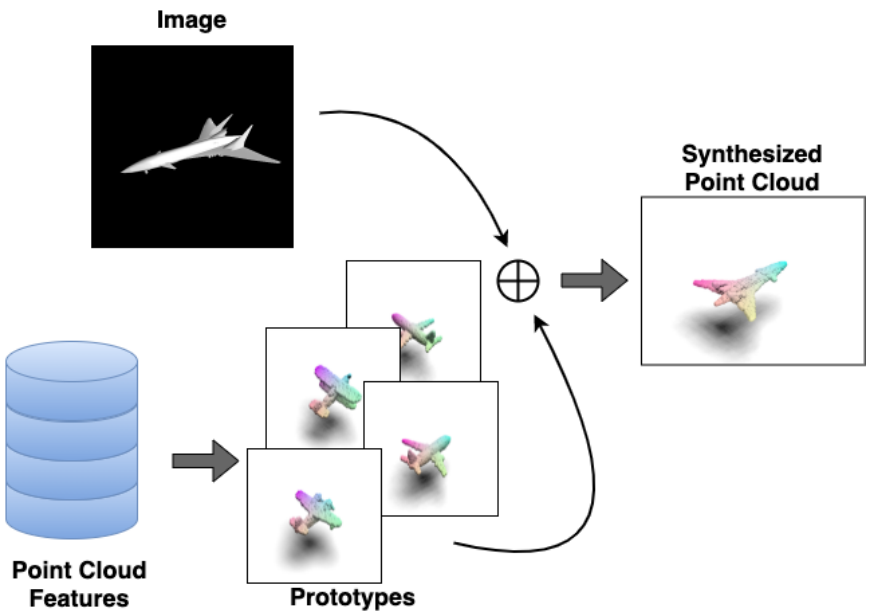

Figure 3.1. Combining 2D image features and 3D prototypes.

$S = \{p_i\}_{i=1}^{N}$, where N denotes its cardinality and p_i is a point in the 3D Euclidean space with coordinate (x_i, y_i, z_i). Based on our observation and existing literature, we found that $N = 2048$ is sufficient to preserve the major structure of a given 3D object (Chang et al., 2015; Achlioptas et al., 2018).

A view image can be considered as a 2D projection of a 3D shape, while it contains limited information about its source. Apparently, such missing information is crucial for a successful reconstruction. We introduce a set of 3D prototype features $T = \bigcup_{c \in C} \{t_i^{(c)}\}_{i=1}^{K}$ to compensate the information loss. Here, C is the set of classes, K is the set size, and t_i is a prototype feature derived from a point cloud dataset. Note that we use the same point cloud dataset both phases for simplicity in Fig 3.2, while a comprehensive external resource, like ShapeNet (Wu et al., 2015), is also applicable in the 1st phase. Let I be the input image and $f(\cdot)$ be the predefined image feature extractor, e.g., VGG-16 or ResNet-18. Our goal is to learn a neural network $G(\cdot \| \theta)$ such that the distance between the synthesized point cloud and the ground truth is minimized. The objective is formulated as:

$$\arg\min_{\theta} D(S, G(f(I) \oplus T \| \theta)) \tag{3.1}$$

where $\theta = \{\phi, \rho\}$ denotes network parameter: ϕ is the parameter of the feature extractor and ρ belongs to the manifold decoders. $D(\cdot, \cdot)$ is the distance function and two common choices of this distance metric are Chamfer Distance (CD) and Earth Mover Distance (EMD). Mathematically speaking, CD and EMD between two sets of points are formulated as following:

$$d_{\mathrm{CD}}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2^2 \tag{3.2}$$

$$d_{\mathrm{EMD}}(S_1, S_2) = \min_{\phi : S_1 \to S_2} \sum_{x \in S_1} \|x - \phi(x)\|_2 \tag{3.3}$$

where $\phi : S_1 \to S_2$ is a bijection. Although these two metrics are widely used by different frameworks, each of them has its own concentration during the generating process, thus leads to different 3D shapes when being used as loss function. EMD favors the shapes close to the "mean-shape" of the given category (Fan et al., 2017). Consider a set of airplanes, the model always outputs a cabin will get a better score. In contrast, CD tends to cover all components while leading to a splashy shape that blurs the object's geometric structure. Noted that the sum term in both equations, such computation is expensive and is another hint we should keep the number of points, N, reasonable in practice.

Workflow

Our framework, TDPNet, has two training phases as illustrated in Fig 3.2. The goal of such design is to prepare the point cloud encoder for prototype extraction and to help the decoder gain the ability to reconstruct point clouds from an isolated 3D feature in the 1st phase. We later build the actual image-to-point-cloud pipeline in the 2nd phase, where the real image features and 3D prototypes are combined and decoded.

To be concrete, we train a point cloud autoencoder in the first step. The encoder may either be a pointwise MLP or a convolution-based network (We adopt PointNet (Qi et al.,

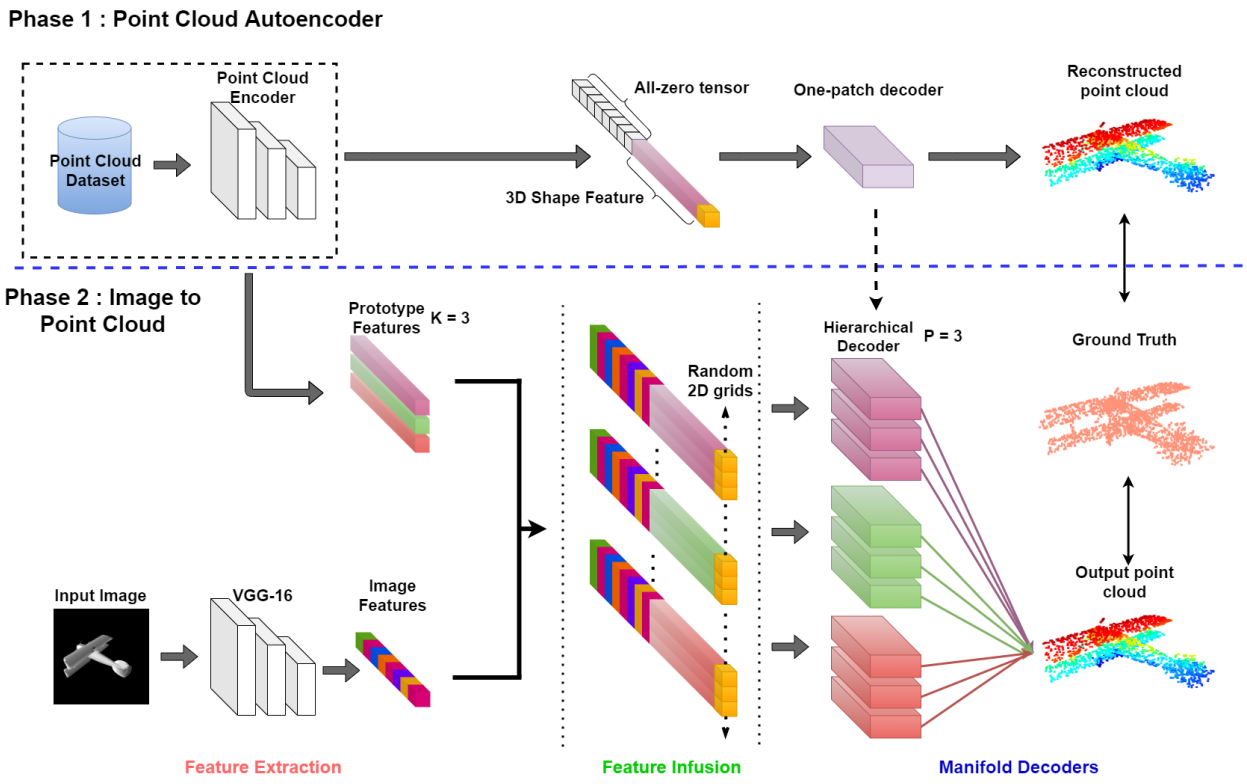

Figure 3.2. Approach overview: A two-phase single-view point cloud reconstruction solution. **a)** We firstly warm-up the network by solving a point cloud self reconstruction problem. Namely, we trained a point cloud autoencoder in this phase. **b)** We build the actual image-to-point-cloud network in the 2nd phase. K prototype features are computed using the trained point cloud encoder and KMeans clustering. Notice that it can be extended to a multi-class version by repeating this operation every class. We then infuse the image feature and prototypes with random 2D grids and feed them to a hierarchical decoder to construct the final point cloud. The decoder has K MLP clusters and each contains P one-patch decoders ($K = 3, P = 3$ in this figure).

2017) in this work). The one-patch decoder is a simple MLP (1538-512-256-128) comprised of ReLU non-linearities on the first three layers and tanh on the last output layer. This decoder will be later used to initialize our hierarchical decoder. Finally, since the image feature I_{feat} is not available in the current stage, we mask out that part with an all-zero tensor. The loss function of this autoencoder may either be CD or EMD.

For the 2nd phase, let's start from a simple scenario, where all inputs are from the same category, e.g., airplane. We need to generate K prototype features with the trained point cloud encoder. A clustering algorithm, such as KMeans, is applied to obtain K clusters. We then initialize the prototype features by the centroid of each cluster. Notice that we need to repeat this operation for every category when facing a general multi-class problem setting. The clustering strategy is suitable mainly because the embeddings are informative and discriminative across categories (Khan and McLeod, 2000; Awad et al., 2008; Nessa et al., 2008). Nevertheless, the centroids are insufficient for a realistic reconstruction, we will apply a Froze-Finetune training strategy on those prototypes, which is explained later.

Our hierarchical decoder contains K MLP clusters and every cluster has P one-patch decoders. We used $K = 3, P = 3$ in Figure 3.2. We concatenate the image feature I_{feat} with corresponding 3D prototypes $T^{(c)} = \{t_i^{(c)}\}_{i=1}^{K}$ to obtain K fused vectors. Recall that there are K MLP clusters and each of them will handle one fused vector. Each fused vector is replicated P times and endowed with randomly sampled 2D grids. With this configuration, we allow each prototype to contribute to the final result democratically and dedicated to different local regions. In the end, all the $K \times P$ patches produced by the decoder are collected onto the final point cloud. The pseudocode for both training phases are presented in Algorithm 1 and Algorithm 2, respectively.

3.1.2 Dynamic 3D Prototypes

In this section, we demonstrate how we generate the 3D prototype and why it's important to use a frozen-finetune training schema.

Algorithm 1: Phase 1 Training

input : A point cloud dataset $S* = \{S_i\}_{i=1}^{m}$

```
for Number of training epochs do
    for batch ← 1 to ⌊m/batch_size⌋ do
        Compute 3D features for S_{batch} ;
        Concatenate 3D features with dummy image features and random 2D grids ;
        Generate Ŝ_{batch} from the fused vector;
        Compute d_CD(Ŝ_{batch}, S_{batch}) ;
        Update the network ;
    end
end
```

Algorithm 2: Phase 2 Training

input : A paired image & point cloud dataset $D = \{I_i, S_i\}_{i=1}^{n}$

```
Generate K prototypes with KMeans;
Initialize all MLPs with the 1st phase decoder;
for Number of training epochs do
    if epoch < frozen_period then
        Froze the prototype
    else
        Activate prototype tuning
    end
    for batch ← 1 to ⌊n/batch_size⌋ do
        Compute 2D features from I_{batch} ;
        Concatenate 2D features with 3D prototypes and 2D random grids ;
        Generate Ŝ_{batch} from the fused vector ;
        Compute d_CD(Ŝ_{batch}, S_{batch}) ;
        Update the network ;
    end
end
```

We initialize the 3D prototypes, $T = \bigcup_{c \in C} \{t_i^{(c)}\}_{i=1}^{K}$, by the KMeans centroids of a collection of point cloud features. Recall that we trained a point cloud AE in the 1st phase and initialized our hierarchical decoder with the one-patch decoder. Indeed, a multiple-patches decoder is acceptable in the point cloud AE, whereas the performance won't be impaired heavily and it could cause negative effects to the initialization. To reduce the requirement of computational

resources and minimize the training time, we stay with the one-patch decoder. Since the clustering algorithm won't change the feature space, we conclude that the centroid captured meaningful information and can be decoded by the one-patch decoder. Examples of extracted prototypes are presented in the experiment section. The one-patch decoder is capable of reconstructing a complete point cloud without any image feature. In other words, the network learns the mechanism to incorporate the 2D features with 3D prototypes in the 2nd phase. The prototypes seem to be random noise and lead to model collapse provided that 1st phase does not exist (Tu et al., 2008; Mejjati et al., 2018), yet it's also not advisable to keep the prototype untouched during the training. For example, an external dataset provides rich shape information while its underlying distribution may not be consistent with the prototype distribution of the 2nd phase.

To overcome aforementioned problems, we propose to froze the prototypes for first few epochs. UAGAN (Mejjati et al., 2018) embraced this strategy to balance the generator and the discriminator in a GAN. The idea behind this operation is general and intuitive: mode collapse is caused by the joint training of one or more auxiliary components. The alleviation of it is allowing update parameters for only one component in the early stage. With the same idea, We froze the prototype for the first 30 epochs and allow them to be fine-tuned in the rest epochs, thus they can capture the correct information in the target dataset.

3.1.3 Hierarchical Manifold Decoder

Following the AtlasNet (Groueix et al., 2018) convention, generating a point cloud can be considered as generating a surface of a 3D shape. The surface (shape) of a 3D object is a differentiable 2-manifold that embedded in the ambient 3D Euclidean space: $M^2 \in \mathbb{R}^3$. A point cloud is considered as a sampled discrete subset of the surface $S = \{p_i \in M^2 \cap \mathbb{R}^3\}$.

Before we dive into the reconstruction process, let us first start with some basic concepts (Zhao et al., 2019):

Definition 1. ***Diffeomorphism*** *is an invertible, differentiable map between two differentiable surfaces.*

Definition 2. *Consider an open set $U \in \mathbb{R}^2$. A* ***chart*** *C is a diffeomorphism $C : M^2 \rightarrow U \in \mathbb{R}^2$ that maps an open neighborhood in 3D space to its 2D embedding.*

Definition 3. *Given a chart C, let $\Psi \equiv C^{-1} : \mathbb{R}^2 \rightarrow M^2$ be the inverse of this chart. Ψ is called a* ***parameterization****.*

Definition 4. *A set of charts with images covering the 2-manifold is called an* ***atlas****: $A = \cup_i C_i(p_i)$*

With these definitions, we conclude that a 2D point set can be deformed to a surface with a parameterization Ψ. In the other words, we are not learning an exact mapping from the hidden vector to a point set $\hat{S}$, but trying to find function(s) $\Psi(U|\rho)$ to generate the 2-manifold, such that $\Psi(U|\rho) \approx S$. ρ is a lower-dimensional parameterization of these functions such that $|\rho| < |S|$.

It has been proved that "Given that C^{-1} exists, arbitrary 3D surfaces can be reconstructed if ψ is approximated by a 3-layer MLP" (Groueix et al., 2018). Based on this theorem and the universal approximation theorem (Csáji et al., 2001), we are able to state that a point cloud S can be universally reconstructed up to a precision ϵ via an MLP with H hidden units.

With these definitions and theorems, previous point cloud decoder networks can be categorized based on their architecture. As presented in Figure 3.3, PointNet (Qi et al., 2017) could be extended to an image-to-point-cloud network naturally by swapping the point cloud encoder with an image feature extractor, and replace the FC-layer with an MLP decoder. PointSetNet (Fan et al., 2017) introduces a MoN loss and improves this architecture by adding a deconvolutional branch and hierarchically combining the output from FC-branch into the final result. However, both of them lack the grid structure and their decode functions depend upon a single latent feature. In other words, these two frameworks have the assumption

$U = \emptyset$. AtlasNet (Groueix et al., 2018) is an advanced version of FoldingNet (Yang et al., 2018). They shared the same intuition of manifold deformation, whereas FoldingNet restricts itself to one manifold, and AtlasNet deforms multiple 2D grids with MLPs.

Although AtlasNet performs very well in the point cloud self-reconstruction task, it assumes that 2D features and 3D features have the same impact on the result point cloud, which is not true in practice. Our framework addresses this problem by combining 2D features and 3D prototypes together. Therefore, our framework is a generalization of AtlasNet, which can be obtained by setting all the prototypes to zero ($T = \emptyset$). Noted that our framework contains one MLP cluster per prototype, thus a prototype can affect several regions if desired.

3.1.4 Implementation

We used a PointNet (Qi et al., 2017) as our point cloud encoder, which is presented in Fig 3.4. Generally speaking, it contains two Conv1D blocks, one for 3D transformation (orange block) and one for hidden space transformation (blue block).

The one-patch point cloud decoder is an MLP (1538-512-256-128) and the first three layers share ReLU non-linearities and the last output layer uses tanh (Groueix et al., 2018). Note that the MLP decoders are implemented using Conv1D layer for efficiency.

Our image feature encoder is just a VGG-16 with batch normalization layer. In order to adjust the network for better performance, we fine-tuned the network by fixing the first few Conv2D layers and allowing only last three Conv2D layers to be updated during the training.

For the hierarchical decoder, which contains K MLP clusters and each cluster contains P one-patch decoders. Assuming we are trying to generate a point cloud with $N = 2048$ points, each one-patch decoder handles $2048/(K \times P)$ points.

For the 1st phase training, we used an ADAM optimizer with an initial learning rate of 1e-3, $\beta = (0.5, 0.999)$ and a batch size of 32. We still use the same ADAM optimizer in the second phase but set the initial learning rate as 1e-4. Moreover, we applied a step learning

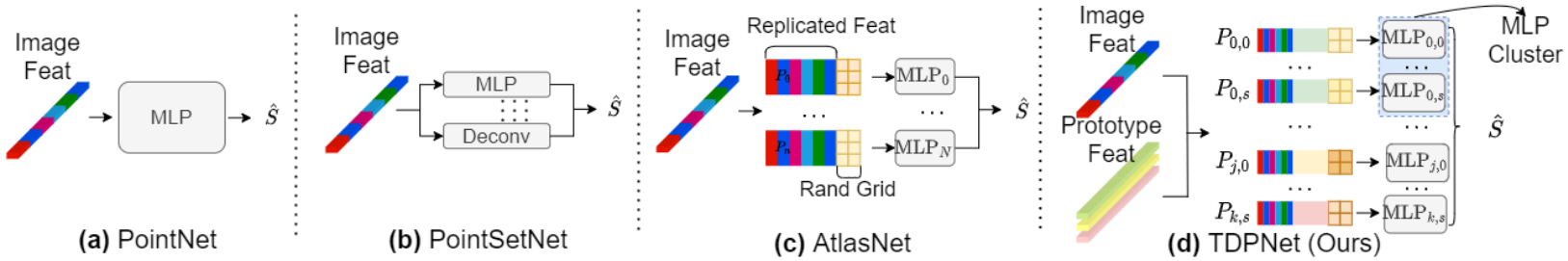

Figure 3.3. Comparison of four SOTA image-to-point-cloud decoders. PointNet and PointSetNet both use a single image feature and have no surface assumption, while PointSetNet is able to generate multiple plausible shapes thanks to the deconv branch and MoN loss function. The AtlasNet tries to deform multiple 2D grids onto local 2-manifold but they still stay on the single image feature. Our TDPNet is capable of fusing 2D and 3D information (K prototype features). Each prototype controls an MLP cluster and every cluster contains P MLP components, each of which can fold a distinct 2D grid onto a specific local point set.

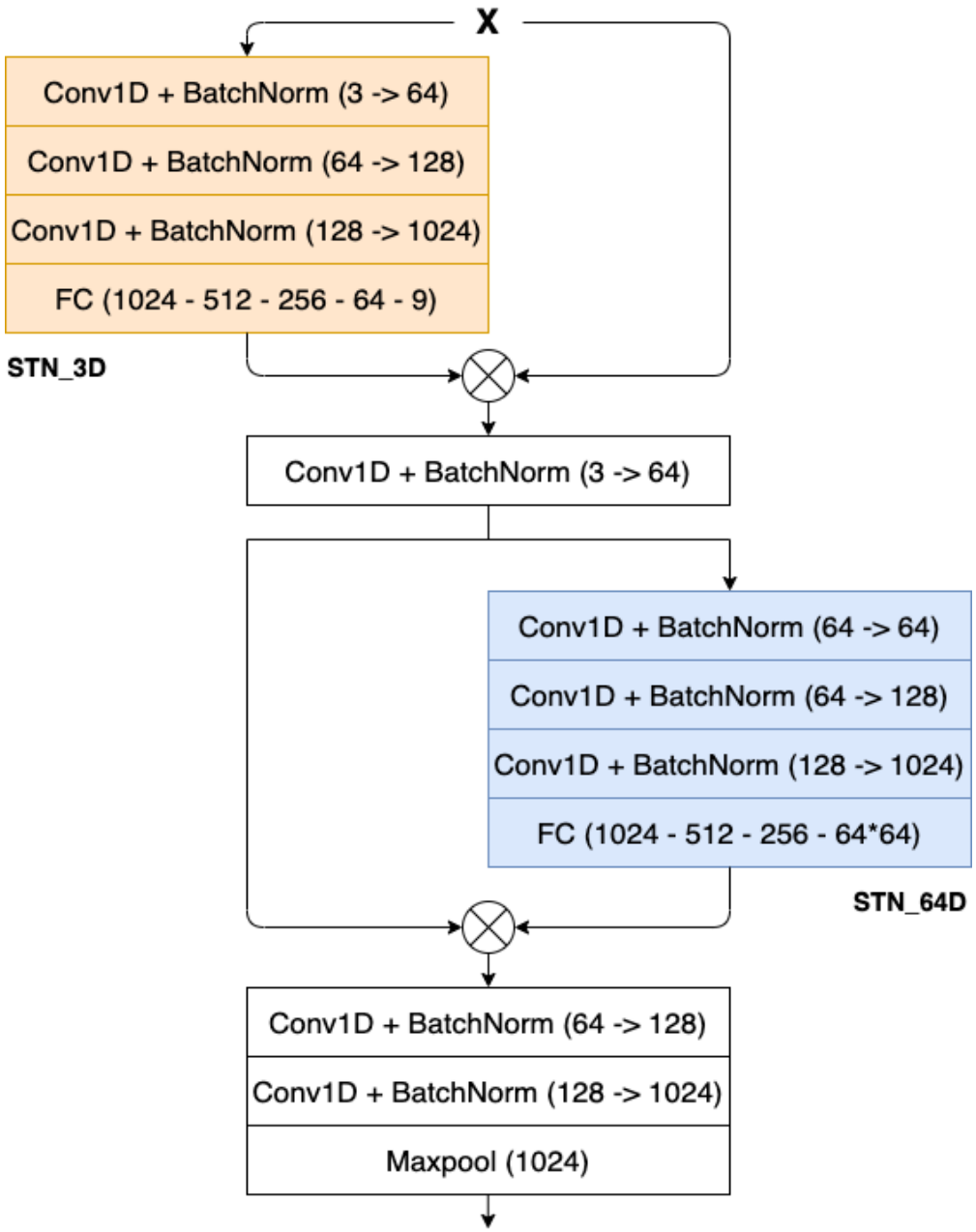

Figure 3.4. Overview of the PointNet point cloud encoder architecture. The input X is a batch point clouds with shape Batch $\times$ 2048 $\times$ 3 and the output is a feature tensor with shape Batch $\times$ 1024. ReLu is the activation function for all Conv1D layers.

rate scheduler to further stabilize the training process, which decrease the learning rate at 30, 60, 90 epochs with $\gamma = 0.5$. We arranged 100 epochs per training stage and froze the prototype for 30 epochs.

3.2 Evaluation

We evaluate our method quantitatively and qualitatively on different challenging tasks, such as single category image-to-point-cloud generation, multiple category image-to-point-cloud generation, and multiple plausible shapes generation.

3.2.1 Experiment Setting

Two datasets are used for the evaluation of this method: ModelNet (Wu et al., 2015), and ShapeNet (Chang et al., 2015). For the ModelNet dataset, we borrow the processed data from MVCNN (Su et al., 2015), which contains 4,899 CAD models across 10 categories and each model is accompanied by 12 2D projections. Regarding ShapeNet, we sampled 13 categories, which totally contains 21,439 CAD models. We then render 12 views of each 3D shape based on the Blinn-Phong shading formula with a black environmental map (Blinn, 1977). The single RGB image of each CAD model in both datasets is chosen from corresponding 12 2D projections randomly. Both datasets are divided into a 80/20 train/test split randomly.

Assume each dataset is denoted as $\{v_i, S_i\}_{i=1}^{N}$, where v_i represents the single-view image and S_i represents the corresponding point cloud. Before the training, the input point clouds are aligned to a common ground plane and size normalized. Specifically, we need to apply a rotation matrix to the point cloud based on each category. If the number of points in the source file is larger than 2048, a random sampling process is applied. For data augmenation, we adopted a rotation matrix with random small angles, $\theta \in [0, \pi/120]$, and a random jitter matrix in $[0, 0.02]$ to the calibrated point cloud.

As mentioned above, we randomly chose one image from the 12 2D projections as the training sing-view image. Based on the image size and data source, we center cropped the image to 550×550 and resized it to 224×224 for all ModelNet categories; All ShapeNet images are first center cropped to 256×256 and then resized to 224×224.

We compare the proposed TDPNet with three SOTA frameworks. PointSetNet (Fan et al., 2017), AtlasNet (Groueix et al., 2018) and PointFlow (Yang et al., 2019). Only the first two methods claim that they have the capacity to generate the point cloud from a single image. Nevertheless, PointFlow solved the point cloud reconstruction task from the perspective of statistics and achieved promising numerical results. Thus, we include this method to study its capacity for the image-to-point-cloud task. For a fair comparison, all the images features are extracted by a VGG-16 and we provide an additional run of AtlasNet with 32 patches, which is equal to the maximum number of MLP decoders in our framework, $K = 8, P = 4$. All quantitative results are the average of 10 runs.

We evaluated the synthesized point cloud by comparing it to ground truth shapes using two criteria: Chamfer Distance and Earth Mover Distance. Formulas and physical meanings of these two criteria are presented in Equation 3.2 and 3.3, respectively. As we will show in later sections, although these two numerical metrics have certain limitations, they unveil different insights to the performance of all models (Yang et al., 2019). In addition, we evaluate the model performance through Jensen-Shannon Divergence (JSD) (Achlioptas et al., 2018) suggested in PointFlow. Noted that previous research suggest Coverage (COV) and Minimum Matching Distance (MMD) as the performance criteria as well, but they are inadequate in our problem setting as the correspondences between images and cloud points are already known. Let S_g be the set of generated point clouds and S_r be the set of ground truth point clouds with $|S_r| = |S_g|$, the JSD are computed between two marginal point distributions:

$$JSD(P_g, P_r) = \frac{1}{2} D_{KL}(P_r \| M) + \frac{1}{2} D_{KL}(P_g \| M)$$

where $M = \frac{1}{2}(P_r + P_g)$. P_r and P_g are marginal distributions of points in the ground truth and generated sets.

3.2.2 Single Category Point Cloud Generation

Recall that our method requires label information to arrange correct 3D prototypes for the generation purpose. To conduct a fair comparison with baselines, we did two sets of experiments to justify the effectiveness of our method. We first evaluate all methods in a single category setting. Namely, the training data and test data are from the same class, so the label offers no extra information. Table 3.1 and Table 3.2 show the results in such setting. It shows that for single view reconstruction, the proposed method consistently achieves better CD and competitive EMD in every categories. Additionally, we can see that our approach is significantly better than AtlasNet with the same number of decoders.

PointSetNet and PointFlow are two extremes in this task. PointSetNet performs moderately in CD but has intolerable EMDs. PointFlow achieves amazing EMDs with unstable CD. Considering the visualization in Figure 3.5 and Figure 3.6, the results reveal the shortcomings of these two metrics. CD favors a splashy result that covers more regions, whereas EMD prefers a "mean-shape" that roughly matches every instance in a given category (Fan et al., 2017; Yang et al., 2019). Based on these observations and previous literatures, we suggest that "a model is better if it has better CDs and moderate EMDs". Adequate EMD guarantees that the result is in the right category and small CD make sure all regions are recovered properly to the correct shape.

3.2.3 Multiple Category Point Cloud Generation

Our framework is capable of solving a more general multiple-class problem. We evaluate all methods in an all categories setting, which means the training set and test set contain data with mixed labels. Table 3.3 presents the result in this setting. Although the performance of

Table 3.1. Single-View Reconstruction (per category) for ModelNet dataset, trained on each category. The results of each framework are reported in format "CD / EMD". Chamfer Distance is multiplied by 10^3 and Earch Mover Distance is multiplied by 10^2 for better visualization. Both metrics are computed on 2048 points. Best results are bolded.

	PointSetNet	PointFlow	AtlasNet 1 Patch	AtlasNet 32 Patches	TDPNet K4P4	TDPNet K8P4
Airplane	6.48 / 36.63	7.96 / **14.47**	6.38 / 21.33	5.94 / 21.22	**5.40** / 19.42	5.44 / 17.13
Bathtub	13.16 / 53.35	33.34 / 21.18	11.36 / 20.55	12.06 / 14.94	10.64 / 14.93	**9.54** / **14.96**
Bed	11.80 / 42.49	10.07 / 15.09	10.14 / 19.28	9.16 / 32.87	7.80 / 13.39	**7.27** / **13.45**
Chair	14.81 / 42.14	11.16 / **15.35**	11.03 / 22.16	9.47/ 16.92	9.86 / 17.43	**8.74** / 17.55
Desk	18.75 / 47.43	28.83 / **23.63**	21.14 / 32.87	21.67 / 34.87	**16.18** / 27.32	18.59 / 31.04
Dresser	18.89 / 55.88	12.15 / 15.52	13.11 / 17.58	10.35 / 14.39	**9.88** / **14.27**	10.18 / 14.71
Monitor	16.49 / 43.91	10.88 / **14.82**	12.88 / 21.59	11.38 / 18.33	10.51 / 16.06	**10.05** / 16.41
Sofa	12.56 / 45.56	9.56 / 14.86	8.66 / 17.03	8.09 / 16.26	**7.59** / **14.36**	8.11 / 15.10
Table	15.46 / 43.69	9.72 / **14.94**	10.49 / 18.45	8.06 / 16.54	7.97 / 15.91	**7.48** / 16.11
Toilet	13.88 / 45.85	12.78 / **16.12**	9.87 / 19.89	9.39 / 21.38	**8.92** / 19.36	9.12 / 20.02

Table 3.2. Single-View Reconstruction (per category) for ShapeNet dataset, trained on each category. The results are organized with the same format of Table 3.1. "AtlasNet 1 patch" and "TDPNet K4P4" are dropped based on their performance.

	PointSetNet	PointFlow	AtlasNet P32	TDPNet K8P4
Airplane	3.36 / 34.71	4.12 / 12.17	2.82 / **11.39**	**2.34** / 13.85
Bowl	37.70 / 50.86	22.38 / **14.99**	17.80 / 15.89	**15.59** / 15.45
Camera	34.27 / 46.08	29.16 / **22.94**	21.41 / 32.63	**16.96** / 29.30
Car	8.63 / 52.39	8.42 / 11.50	4.42 / 11.39	**4.20** / **11.18**
Cellphone	6.87 / 39.90	5.77 / **9.54**	4.25 / 11.36	**3.71** / 9.68
Chair	6.35 / 45.15	8.95 / 14.72	6.67 / **13.81**	**6.32** / 14.87
Clock	24.89 / 58.15	17.34 / 18.04	12.46 / 18.57	**11.06** / **16.03**
Faucet	24.32 / 47.56	12.12 / **16.96**	8.49 / 29.22	**6.99** / 28.44
Jar	44.03 / 53.44	14.00 / **15.75**	11.56 / 16.76	**10.90** / 16.83
Monitor	14.75 / 39.56	8.11 / 14.37	6.74 / 13.65	**6.13** / **13.48**
Mug	15.78 / 60.05	13.58 / **13. 25**	**9.46** / 14.18	10.03 / 13.85
Printer	28.88 / 40.08	20.72 / 19.50	17.31 / 19.87	**16.03** / **19.34**
Rocket	7.14 / 32.48	8.52 / 14.47	3.94 / 13.54	**2.64** / **12.39**

our method downgrade slightly comparing to the single-category tests, we still achieve better scores in most categories. Moreover, by incorporating the label information and enjoying the copious training data, our model gets improved EMD scores, which outperforms all competing methods. An interesting observation here is that some classes, e.g., sofa and dresser, got better results compared with its single-category version. We suggest that it is because of the insufficient training data in the single-category setting.

In addition, we provide the results regrading JSD of each category in this multiple category setting in Table 3.4. JSD score measures the divergence of the marginal point distribution between reconstruction ground truth. Lower JSD indicates that the two distributions are closer and the synthesization is more realistic. Our approach shows its robust performance and outperforms all baselines in most categories. Based on these experimental performance, we conclude that point cloud generation quality can be significantly improved by incorporating label information and latent 3D prototypes.

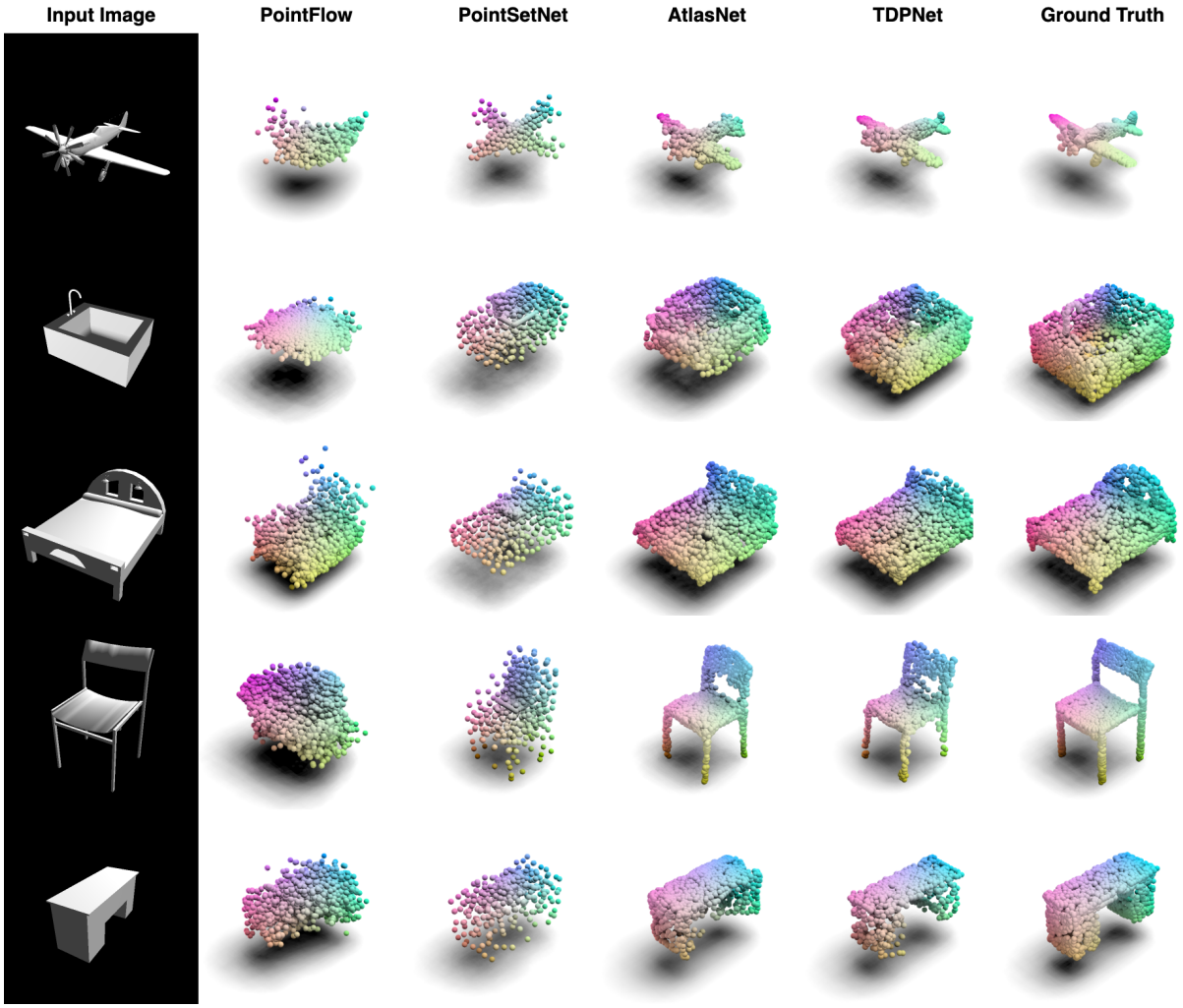

Figure 3.5. Examples of qualitative comparison (ModelNet) among different method. From left to right: Input image, PointFlow, PointSetNet, AtlasNet, TDPNet (Ours) and Ground Truth.

3.2.4 Generating Multiple Plausible Point Clouds

Due to the information loss introduced by the projection, multiple reconstructions are expected for a single image. Given the same input image, the random sampled coordinates from the 2-manifold naturally allows prediction of different shapes. This model behavior is very valuable because of the ambiguous behavior of 2D to 3D construction (Sung et al., 2018).

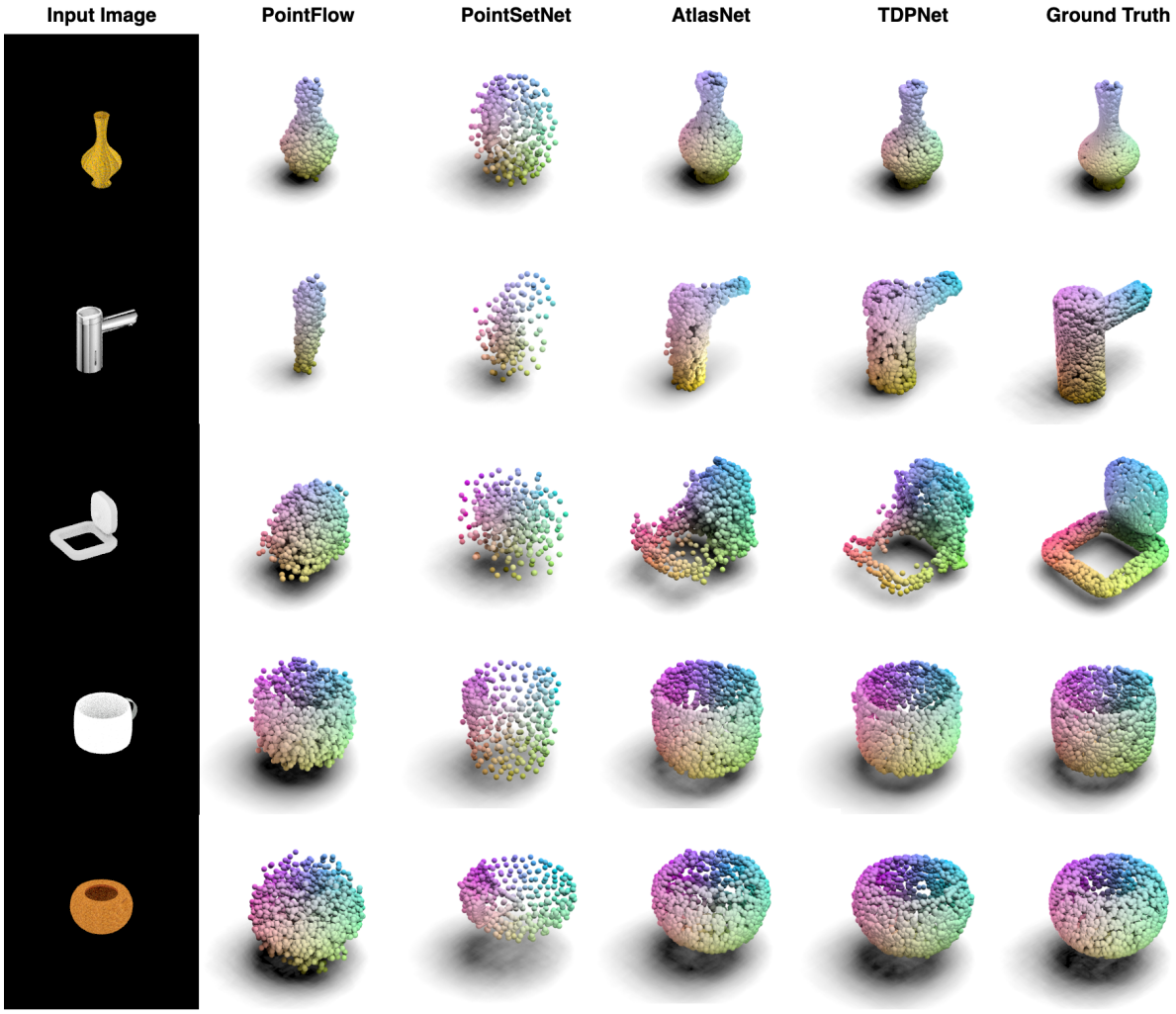

Figure 3.6. Examples of qualitative comparison (ShapeNet) among different method. From left to right: Input image, PointFlow, PointSetNet, AtlasNet, TDPNet (Ours) and Ground Truth.

Figure 3.7 shows examples of a collection of predictions given one image. We observed that our network can reveal its uncertainty about the shape or the ambiguity in the input.

3.3 Ablation Studies and Discussion

In this section, we will further investigate the impact of different network components and discuss the limitation of the proposed method.

Table 3.3. Single-View Reconstruction (per category) for ModelNet, trained on all categories. The results are in format "CD / EMD" and they are scaled by 10^3 and 10^2, respectively.

	PointSetNet	PointFlow	AtlasNet P32	TDPNet K8P4
Airplane	20.92 / 42.54	14.28 / 24.52	8.03 / 33.56	**5.68** / **21.74**
Bathtub	44.83 / 53.05	21.11 / 20.25	16.17 / 37.27	**8.73** / **18.17**
Bed	18.77 / 49.82	12.37 / 17.65	8.11 / **15.08**	**7.50** / 17.90
Chair	23.12 / 44.99	17.54 / 27.41	14.10 / 27.52	**9.52** / **23.85**
Desk	27.79 / 49.06	31.72 / 27.97	21.32 / 41.49	**16.61** / **27.84**
Dresser	54.45 / 56.65	12.15 / **15.52**	17.64 / 22.37	**9.74** / 20.35
Monitor	31.83 / 50.88	15.05 / 24.22	13.08 / 20.76	**9.09** / **20.01**
Sofa	16.59 / 50.19	14.09 / 18.97	10.10 / **16.88**	**7.64** / 20.30
Table	22.45 / 47.88	10.91 / 20.77	9.23 / 25.74	**7.03** / **18.36**
Toilet	23.29 / 49.67	17.14 / **27.28**	9.89 / 30.17	**8.96** / 27.75

Table 3.4. JSD score of Single-View Reconstruction (per category) for ModelNet, trained on all categories.

	PointSetNet	PointFlow	AtlasNet P32	TDPNet K8P4
Airplane	0.8863	0.2204	0.2378	**0.2039**
Bathtub	0.8712	0.1623	0.1721	**0.0796**
Bed	0.8591	**0.0539**	0.1727	0.0988
Chair	0.8839	0.2142	0.2106	**0.1953**
Desk	0.8614	**0.1545**	0.1721	0.1684
Dresser	0.8885	0.1669	0.1185	**0.0901**
Monitor	0.9022	0.3785	0.1323	**0.0942**
Sofa	0.8491	0.1236	0.1212	**0.0943**
Table	0.8551	0.1582	0.1084	**0.0941**
Toilet	0.8706	**0.1405**	0.1936	0.2066

Table 3.5. Chamfer Distance measured on ModelNet-airplane with different hyper-parameter setting. The result are multiplied by 10^3 for better visualization.

	P=1	P=2	P=4	P=8
K=2	6.41	6.28	5.70	5.63
K=4	6.15	6.09	5.40	5.66
K=8	6.08	5.82	5.44	5.41
K=16	5.73	5.41	5.25	5.07

3.3.1 Contribution of Prototypes and Decoders

We report the result of different combination of K and P in Table 3.5. Notice how our approach generally improves as we increase the number of prototypes and MLP decoders.

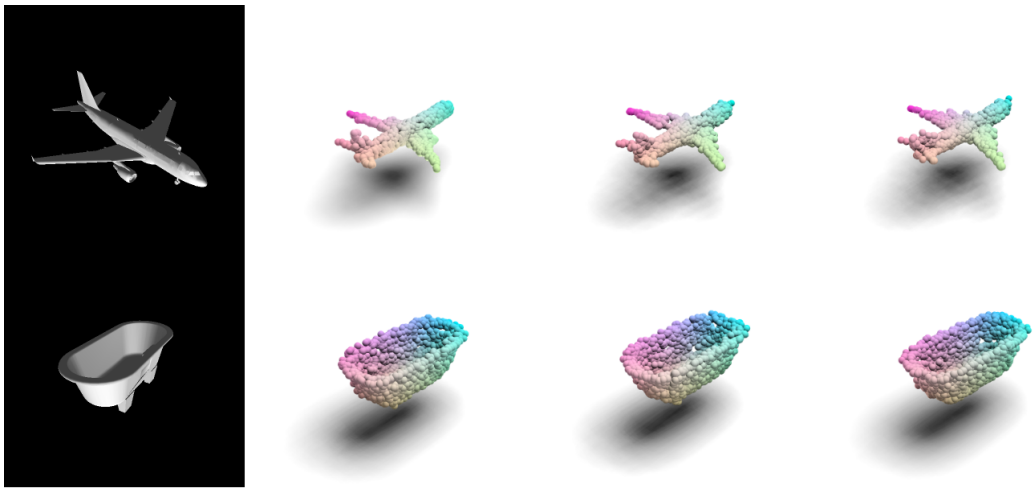

Figure 3.7. Multiple predictions for a single input image. Note that the input view can be a 2D projection from a different angle, while we can still reconstruct the 3D shape correctly.

Table 3.6. Chamfer Distance ($\times 10^3$) measured on 2 ModelNet categories with different training strategies. We adopt configuration of $K = 8, P = 4$ in this experiment.

	Froze	Finetune	Froze-Finetune
Airplane	6.12	5.76	5.44
Bathtub	11.37	10.84	9.54

Another interesting observation is that our approach usually gains more benefits from the increase of prototypes compared to the increase of MLP decoders. In other words, the 3D prototypes provide more valuable information compared with the stacking of decoders. Our approach consistently outperforms AtlasNet when the number of decoders is equal ($K \times P = 32$), which further justified the effectiveness of combining 2D and 3D features. Finally, we observed that adding extravagant decoders doesn't necessarily improve the model performance.

3.3.2 Frozen-Finetune Training

In this section, we evaluate the effectiveness of the Frozen-Finetune training. Table 3.6 shows the quantitative results. The performance of Frozen and Finetune are approximately the same since the network may recognize the former one as constants and the finetuned version

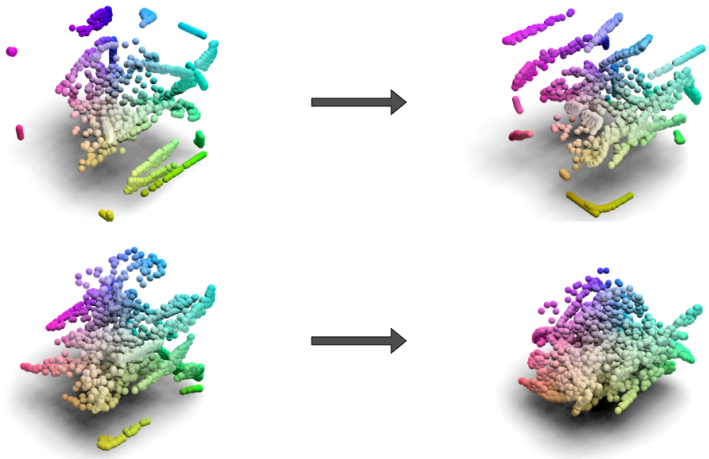

Figure 3.8. Sampled prototypes on ModelNet-bathtub. Left are the initial centroids and Right are the finetuned prototypes.

as random noise. On the other hand, our approach is capable of avoiding mode collapse and utilizing the 3D shape information.

Figure 3.8 visualizes two samples of prototypes. We combine the prototype with a dummy image feature and feed it to the 1st phase decoder. Although this may not be the optimal visualization because we trained the prototype features to incorporate with "real" image features, we can still find that the finetuned prototype looks closer to the category mean-shape, whereas the frozen version tends to be more sparse in the space.

3.3.3 Discussion

Our results have limitations that lead to many open questions and perspectives for future work. First of all, we adopted KMeans as the prototype aggregate strategy. Although it performs well in our experiment, it would be interesting to study the behavior of different clustering methods or different generation methods, such as the prototypical network (Snell et al., 2017). Second, we are focusing on the object-centered point cloud synthesization, yet extending to entire real scene point clouds is very attractive. Third, the infusion strategy we used for the 2D image feature and 3D prototypes is straightforward. Even we achieved a realistic

reconstruction, it would be interesting to see some sophisticated feature infusion methods. For instance, attention mechanism (Guo et al., 2020) and GAN-based metric learning (Musgrave et al., 2020). Finally, an open question is how to define a good performance metric that is consistent with humans' observation. As we can see in this chapter, PointFlow (Yang et al., 2019) achieves extraordinary EMD scores, whereas the qualitative results are not promising. It would be an interesting future direction to solve these problems.

CHAPTER 4

POINT CLOUD GENERATION VIA LEARNABLE 3D PRIORS

4.1 Approach

Our previous method, TDPNet (Lin et al., 2021), incorporates 3D prior knowledge by using a collection of latent prototype features. Although it achieves promising results, we found that using KMeans centroids as the prototype features is unstable and sensitive to the hyper-parameters, e.g., K. To address these issues, we further propose a novel single-view point cloud generation framework with **L**earnable **SHA**pe **P**rimitive (L-SHAP), which learns the 3D shape prior information from data and avoids the unstable clustering process. In this chapter, we focus on designing a general framework for the point cloud generation tasks that allow efficient shape primitives learning. Specifically, as shown in Figure 4.1, 3D objects are decomposed into coarse components, whose geometric information is captured by the proposed learnable shape primitives.

This chapter is organized as follows. We will first walk through the whole framework and describe two alternative learnable shape primitives in detail. In the evaluation section, the effectiveness of the proposed method is justified upon two popular benchmarks. Finally, we conduct some ablation studies to investigate the impact of different network components and discuss the limitations of our method.

4.1.1 Framework Overview

We aim to learn a set of shape primitives, as 3D shape priors, to help generate a complete point cloud from a single RGB image. Let $D = \{(I_i, S_i)\}_{i=1}^{N}$ be a collection of image-point-cloud pairs, where N denotes the cardinality of this dataset, I_i and S_i indicate i-th RGB image and point cloud, respectively. Moreover, a point cloud S with M points is formulated as $S = \{p_j\}_{j=1}^{M}$ and each point p_j is represented by its coordinates (x_j, y_j, z_j) in a 3D Euclidean

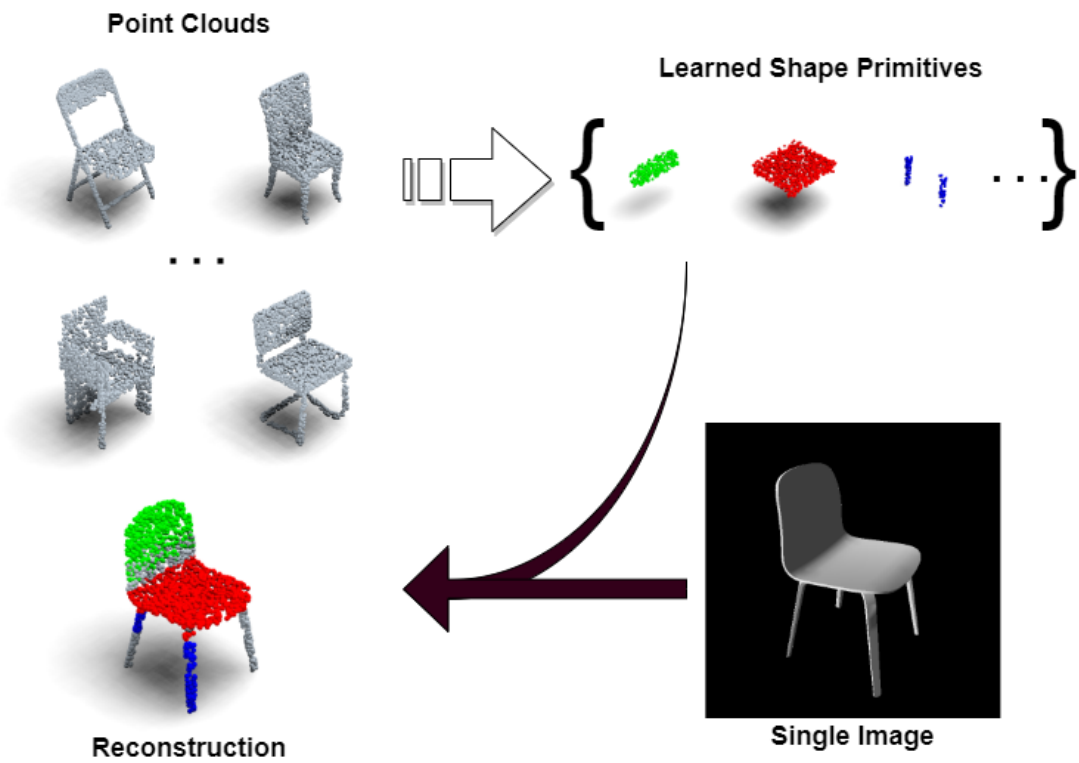

Figure 4.1. Learning shape primitives to encode 3D prior knowledge from dataset. After that, they are incorporated with a given image to reconstruct the target point cloud.

space. Although a point cloud can conceptually have infinite points, we follow the previous chapter and choose $M = 2048$ to balance between the expressiveness and the computational cost (Wu et al., 2015; Lin et al., 2021).

In this work, we consider a point cloud generated from images is formed by a set of adjusted shape primitives. Because of the variety and complexity of 3D shapes, manually defining the structure of shape primitives (Schnabel et al., 2007, 2009; Li et al., 2011; Debnath et al., 2018; Kaiser et al., 2019) impairs their expressiveness, especially for reconstruction tasks. On the contrary, automatically learning the shape primitives from data provides a better representation (Genova et al., 2019). Specifically, we consider each shape primitive as a set of points that encodes the geometric information of a 3D point cloud component. To make the primitive trainable, we follow the convention of AtlasNet (Groueix et al., 2018) and assume that the complex point set is transformable from a simpler one, E_{init}, via a

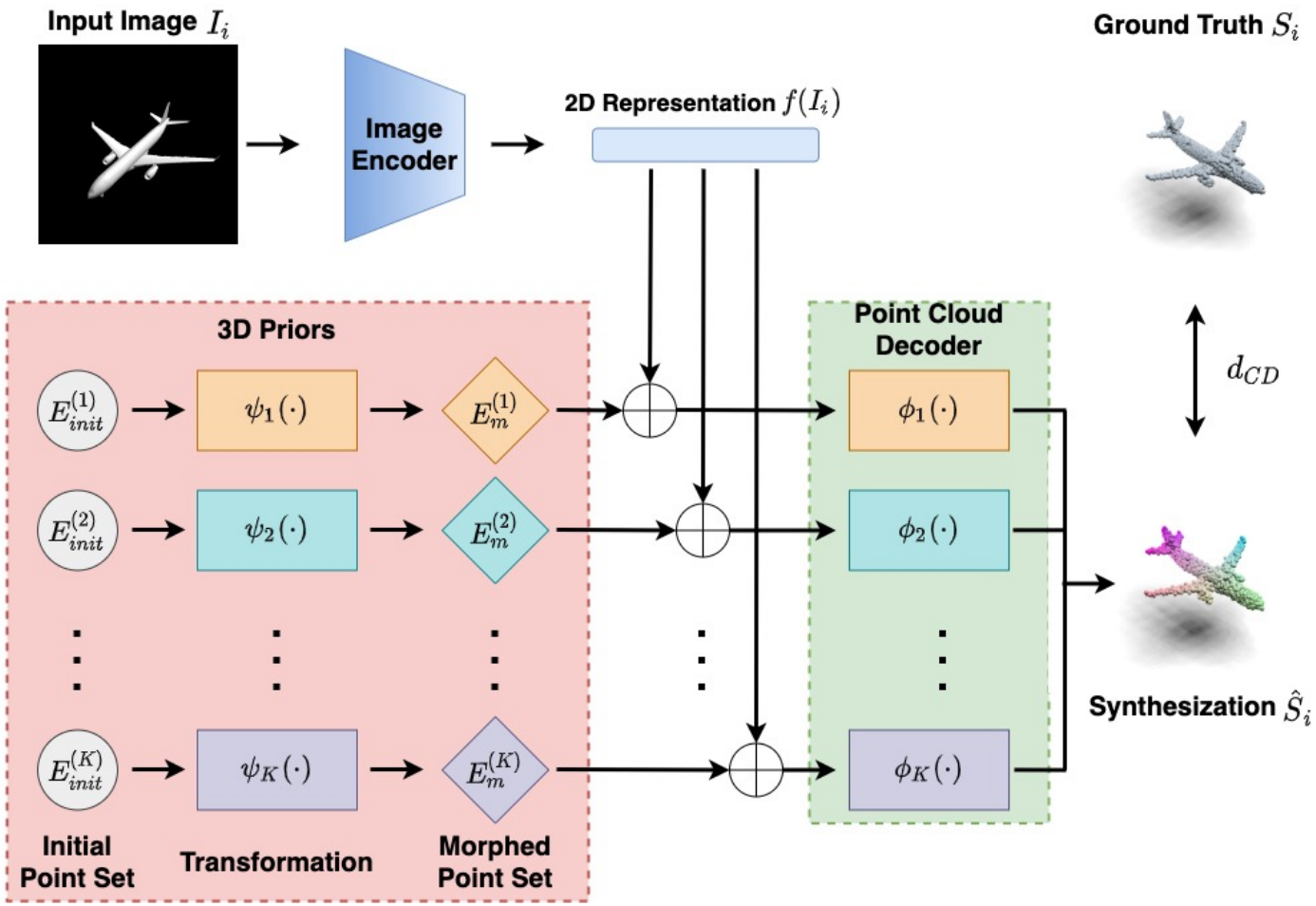

Figure 4.2. Framework overview. For a given image I_i, we first compute its latent representation $f(I_i)$ through an image encoder. K shape primitives are employed to encode the geometric information of shape components. Each contains an initial point set E_{init}, e.g., points sampled from unit square, and a transformation function $\psi(\cdot)$. Later, we endow primitives with $f(I_i)$ and decode them onto the final point cloud $\hat{S}_i$. Chamfer distance between ground truth S_i and synthesization $\hat{S}_i$ is computed to update the network during training.

transformation function $\psi(\cdot)$. Formally, the k-th shape primitives $E_{prim}^{(k)}$ is presented as:

$$E_{prim}^{(k)} = \psi_k(E_{init}^{(k)}) \tag{4.1}$$

The detail of this transformation will be described in Section 4.1.2. After learning the transformations, we can apply these shape primitives to generate a point cloud from a single image. More specific, we consider the process of reconstructing a point cloud from an image as an image-guided primitive transformation process. Let $P = \{E_{prim}^{(k)}\}^K$ be the set of learned shape primitives, a point cloud $\hat{S}_i$ is generated from corresponding image I_i by:

$$\hat{S}_i = \bigcup_K \phi_k(f(I_i) \oplus E_{prim}^{(k)}) \tag{4.2}$$

where $f(\cdot)$ is a predefined image feature encoder, e.g., ResNet-18, $\oplus$ denotes the endow operation (See Figure 4.3), and $\phi_k(\cdot)$ is a primitive-specific transformation function that maps the feature endowed shape primitive to a specific part of the target object. The intuition behind our approach is that if shape primitives are indeed 3D priors, it would be easier for $\phi_k(\cdot)$ to combine 2D information and 3D priors for improving the point cloud generation quality. It's worthy of noting that if the shape primitives were 2D squares, our approach degrades into an AtlasNet (Groueix et al., 2018), and if the shape primitives were KMeans centroids of a point cloud dataset, this equation is identical to TDPNet (Lin et al., 2021).

Our objective is to build a 3D-aware model that learns a collection of parameterized shape primitives and utilizes them to generate a synthesized point cloud $\hat{S}$, such that the distance between $\hat{S}$ and corresponding ground truth S is minimized. Recall that two famous distance metrics between point clouds are Chamfer Distance (CD) and Earth Mover Distance (EMD). Mathematically, CD and EMD between two point clouds S_1 and S_2 are formulated as following:

$$d_{\mathrm{CD}}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2^2 \tag{4.3}$$

$$d_{\mathrm{EMD}}(S_1, S_2) = \min_{g: S_1 \to S_2} \sum_{x \in S_1} \|x - g(x)\|_2 \tag{4.4}$$

where $g : S_1 \rightarrow S_2$ is a bijective function. Each metric contributes differently to the final output. EMD favors the shapes close to the "mean-shape" of the given category (Fan et al., 2017). In contrast, CD tends to cover all components while leading to a splashy shape that blurs the object's geometric structure. As suggested by previous literature (Masud et al., 2009; Groueix et al., 2018; Lin et al., 2021), we choose CD as our loss function and evaluate the performance with both metrics.

4.1.2 Learnable 3D Priors

An RGB image solely contains deficient information for a complex 3D shape, therefore 3D priors are crucial for a realistic single-view point cloud reconstruction. Prior works have shown that a complex 3D shape can be abstracted by simple, yet geometrically informative, volumetric primitives (cuboids) (Masud et al., 2010; Tulsiani et al., 2017; Zou et al., 2017). Inspired by these approaches, we try to learn good primitives for point clouds. Nevertheless, it's not a trivial task to extend cuboids (or other predefined shape primitives) to point cloud domains because cuboids are low-level representations which do not contain shape-specific information. Here, we introduce two types of shape primitives to close the gap.

Continuous Shape Primitives

Analogously to manifold-based methods (Yang et al., 2018; Groueix et al., 2018; Lin et al., 2021), we first assume that each shape primitive has a continuous structure. Specifically, we assume the initial point set $E_{init}^{(k)}$ is a simple continuous point distribution and the transformation $\psi^{(k)}(\cdot)$ maps it into a complex point distribution. These two components jointly encode the crucial 3D prior information from training data.

Further, the aforementioned methods unanimously assume that the initial point sets are differentiable 2-manifolds (surface) and the transformation approximates a diffeomorphism. We can observe that they are a special case of continuous shape primitives, where the initial

point set represents a 2D unit square. By extending the geometric structure from a simple 2D surface to any kind of continuous topological object, we can better capture the information of complex 3D shapes. In addition, the transformed point sets preserve the continuous property, thus any number of points can be sampled from it. In other words, these kind of shape primitives open the gate to dense point cloud reconstruction.

Discrete Shape Primitives

Continuous primitives enable dense point cloud generation, yet they cannot change the topology of the initial structure – this drawback dramatically limits their application for complex shapes. Based on this observation and previous work (Yang et al., 2018), we propose a more flexible "discrete shape primitive".

Concretely, we remove the continuous assumption and model each point set merely as a discrete distribution of points. Without the continuous neighborhood constraint, the transformation function is able to freely adjust the location of output points. Such flexibility significantly boosts the expressiveness of the learnt primitives, yet dense point cloud generation is no longer supported as we dropped the continuous structure.

4.1.3 Architecture and Training Strategy

We now describe the architecture of our framework in detail. As presented in Figure 4.2, our 3D-ware framework adds a shape primitive component to the classical encoder-decoder network structure. To avoid the discrepancy caused by different image encoders, all image encoders in baselines and our method are replaced by a pretrained VGG-16. Note that all layers except the last three convolutional layers in VGG-16 are freezed during training.

Each shape primitive transformation function $\psi(\cdot)$ is implemented by a D-128-128-3 MLP, where D is the dimension of the initial structure (e.g., $D = 2$ if the initial structure is a unit square). For the activation function, we adopts ReLU non-linearities on the first two layers and

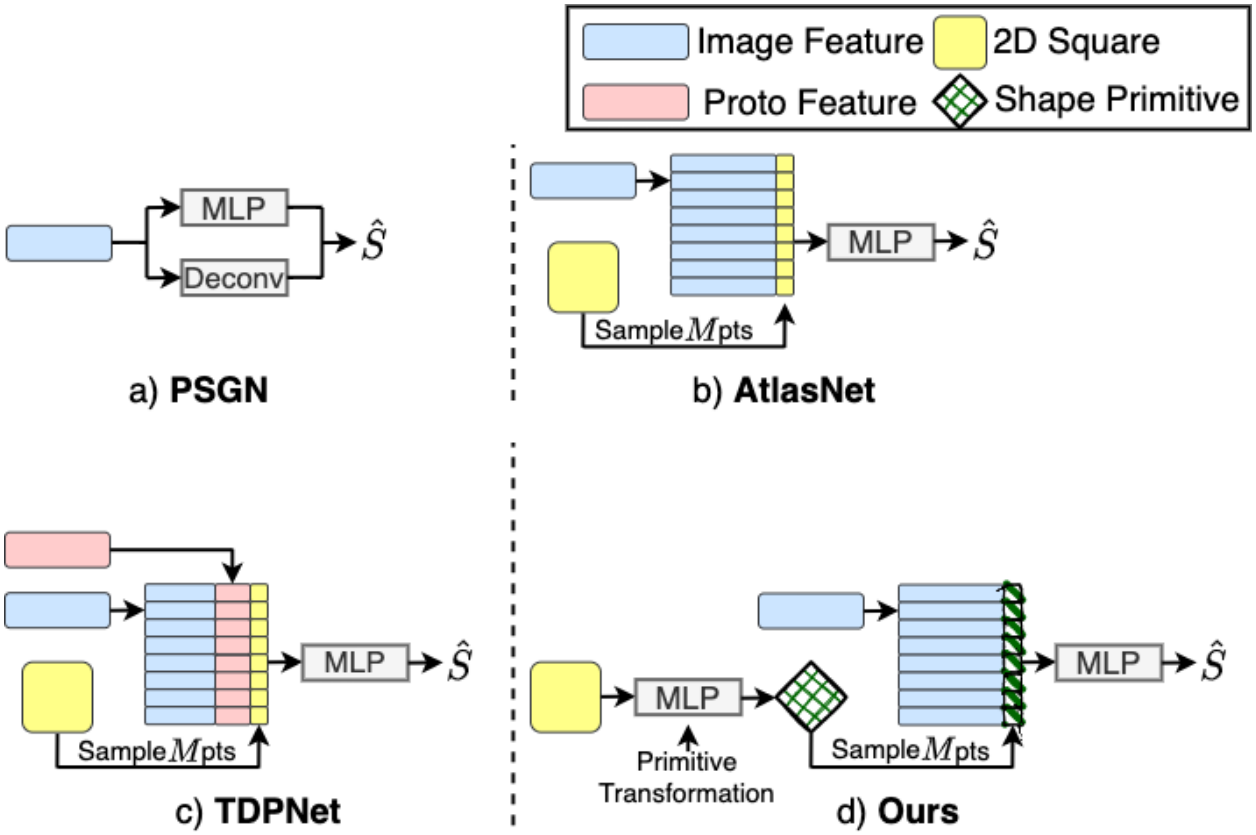

Figure 4.3. Comparison of four point cloud decoders. PSGN directly maps an image feature to a point cloud $\hat{S}$ with M points. The rest methods generate M points by repeating the showing architecture T times, each handles M/T points ($T = 1$ in this plot for simplicity). AtlasNet replicates image feature M times and concatenates them with randomly sampled points from a 2D square. TDPNet extends the pipeline of AtlasNet by adding a 3D prototype feature. In contrast, our method encodes 3D shape information into learnable shape primitive and samples M points from it during inference. Furthermore, our approach is capable of handling various initial point sets (e.g., 3D Gaussian).

tanh on the last output layer. In addition, we tried a shallow 3-layer DGCNN by replacing the FC-layers in MLP with EdgeConv block (Wang et al., 2019) This type of networks explicitly considers the local ambient information of each point during the transformation. Interestingly, both implementations lead to similar performance in the experiments.

For fair comparison, we adopt the hierarchical decoder proposed by TDPNet to decode each endowed shape primitive to a small point set. That is, every $\phi(\cdot)$ is implemented by 4 parallel MLPs (515-515-256-128-3) with ReLU activation in the first three layers and tanh in

Algorithm 3: L-SHAP Training and Inference

input : Dataset $D = \{(I_i, S_i)\}_{i=1}^{N}$, number of points per point cloud M, number of shape primitives K.

```
for Training epochs do
    for i ← 1 to N do
        Compute I_i's latent representation f(I_i);
        Ŝ = ∅ ;
        for j ← 1 to K do
            Sample M/K points from E_init^(j)
            Q ← RAND(E_init^(j), M/K) ;
            Transform points with ψ_j (Eq. 4.1)
            Q' ← ψ_j(Q) ;
            Concat Q' with f(I_i) (Eq. 4.2)
            Q'' = Q' ⊕ f(I_i);
            Transform endowed primitive Ŝ = Ŝ ∪ φ_j(Q'')
        end
        Calculate d_CD(Ŝ, S_i) ;
        Update the network ;
        # NO update during the inference ;
    end
end
```

the last layer. With this configuration, each shape primitive is able to encodes more complex shape component and contributes to the final output democratically. At last, we collect all the points produced by the decoder to form the final point cloud. Algorithm 3 presents the pseudocode for the training and inference process (batch_size = 1).

4.2 Evaluation

In this section, we quantitatively and qualitatively evaluate our approach on three tasks: 1) single category single-view reconstruction, 2) multiple category single-view reconstruction, and 3) dense point cloud generation. Then, we provide the visualizations and contributions of the shape primitives, perform ablation study on different framework components, and investigate the influence of high-dimensional shape primitives in Section 4.3.

4.2.1 Experiment Setting

Follow the same evaluation protocol of TDPNet (Lin et al., 2021), we evaluate our framework on two popular benchmark datasets:

- ModelNet (Wu et al., 2015): We select a 10-category subset of ModelNet with 4899 CAD models (3977/922 train/test split). For each 3D model, we create 12 rendered views by uniformly placing 12 virtual cameras around it. We randomly chose one view as the input RGB image. The ground truth (GT) point clouds are generated via Farthest Point Sampling (FPS) (Qi et al., 2017).

- ShapeNet (Chang et al., 2015): We sampled 3 categories with 14355 3D models (11537/2818 train/test split). Instead of using FPS, each GT point cloud is uniformly sampled from the surface of corresponding CAD model. Input images are generated by the same procedure described above.

We compare our framework with two categories of models: image-only models and 3D-aware models.

- PSGN (Image-only): This is the first learning-based image-point-cloud reconstruction framework. It has no 3D prior and uses a hybrid decoder to generate the point cloud output.

- AtlasNet (3D-aware): It's a pioneer approach that embraces the manifold assumption. It deforms multiple feature-endowed patches onto the target's surface.

- FoldingNet (3D-aware): It generates the output point cloud by deforming an endowed 2D patch twice, which allows the change of topology.

- TDPNet (3D-aware): This is a state-of-the-art 3D-aware method, which generates point clouds by combining the image feature with predefined 3D prototypes. Its decoder is a hierarchical AtlasNet decoder.

For a fair comparison, we choose both Chamfer Distance (Equation 4.3) and Earth Mover Distance (Equation 4.4) to quantitatively evaluate the performance of all methods.

The implementation of our framework is based on Python 3.6 and PyTorch 1.2.0. All experiments are conducted with a Intel i9-9980XE CPU @ 3.00GHz and two Nvidia Quadro RTX 8000 GPU cards. Each input point cloud is aligned to a common plane and normalized into a unit ball. Data augmentation techniques such as random rotation and jitter are applied in the experiment. We center crop and resize all RGB images to a resolution of 224×224. Our model is trained in 200 epochs using an ADAM optimizer with an initial learning rate of 10^{-3} and $\beta = \{0.9, 0.999\}$. If not noted, our method adopts 8 shape primitives, surface transformation and unit square initial structures.

4.2.2 Single Category Point Cloud Generation

Recall that the shape primitives lie at the heart of our approach. They tend to be more shape-specific when facing a single-category scenario and might be more general when dealing with shapes from all categories. To conduct a thorough comparison with baselines, we perform two sets of experiments to verify the effectiveness of our method.

We first evaluate all methods in a single category setting, which means the training data and test data are from the same class. Table 4.1 and Table 4.2 present the results on ModelNet and ShapeNet, respectively. Please note that "AtlasNet P1" is an AtlasNet with an one-patch decoder, while "AtlasNet P32" denotes an AtlasNet with 32 one-patch decoders. "K4" and "K8" in TDPNet indicate the number of 3D prototypical features, each prototypical feature is handled by 4 one-patch decoders. For a fair comparison, our framework adopts 8 shape primitives and utilizes the same decoder architecture as TDPNet. Such configuration minimizes the discrepancy between the decoders of "AtlasNet P32", "TDPNet K8", and our method.

First and foremost, please notice how the results are generally improved as we go from PSGN to our method. Such improvement demonstrates the importance of adopting flexible

Table 4.1. Performance comparison between baselines and our method on ModelNet in the single-category setting. We report the results of each framework in the format of CD (x10^3) / EMD (x10^2). The average performance among all categories is shown in row **AVG**. Both metrics are computed on point clouds with 2048 points, and the best results are highlighted in bold.

	PSGN	FoldingNet	AtlasNet P32	TDPNet K8	L-SHAP (Ours)
Airplane	6.48 / 36.63	6.03 / 24.59	5.94 / 21.22	5.44 / **17.13**	**5.14** / 18.43
Bathtub	13.16 / 53.35	15.91 / 41.53	12.06 / 14.94	**9.54** / **14.96**	10.33 / 15.21
Bed	11.80 / 42.49	10.08 / 18.69	9.16 / 32.87	**7.27** / 13.45	7.73 / **13.16**
Chair	14.81 / 42.14	10.25 / 24.32	9.47 / 16.92	8.74 / 17.55	**8.26** / **16.45**
Desk	18.75 / 47.43	27.47 / 41.88	21.67 / 34.87	18.59 / 31.04	**16.45** / **27.06**
Dresser	18.89 / 55.88	12.47 / 28.22	10.35 / 14.39	10.18 / 14.71	**9.47** / **14.15**
Monitor	16.49 / 43.91	12.38 / 31.64	11.38 / 18.33	10.05 / **16.41**	**9.52** / 17.34
Sofa	12.56 / 45.36	9.75 / 15.97	8.09 / 16.26	8.11 / 15.10	**6.87** / **13.96**
Table	15.46 / 43.69	9.19 / 21.71	8.06 / 16.54	7.48 / 16.11	**6.32** / **14.07**
Toilet	13.88 / 45.85	11.58 / 22.44	9.39 / 21.38	9.12 / 20.02	**7.58** / **16.34**
AVG.	14.15 / 45.09	12.09 / 26.06	10.31 / 20.97	9.29 / 17.64	**8.52** / **16.58**

Table 4.2. Performance comparison between baselines and our method on ShapeNet in the single-category setting. The results are organized in the same format as Table 4.1. AtlasNet P1 and TDPNet K4 are omitted since they are surpassed by their variants.

	Airplane	Chair	Car	**AVG.**
PGSN	3.36 / 34.71	6.35 / 45.15	8.63 / 52.39	6.08 / 44.01
FoldingNet	2.79 / 11.47	9.10 / 29.65	5.05 / 12.41	6.26 / 20.05
AtlasNet P32	2.82 / 11.39	6.67 / 13.81	4.42 / 11.39	4.99 / 12.50
TDPNet K8	2.34 / 13.85	6.32 / 14.87	4.20 / 11.18	4.64 / 13.63
L-SHAP (Ours)	**2.06 / 10.75**	**5.08 / 11.25**	**4.07 / 10.97**	**3.96 / 11.03**

3D priors. Moreover, it is also observed that the proposed method consistently achieves better CD and EMD in almost every category for single-view reconstruction. Most importantly, our approach, on average, provides a better performance than TDPNet, which demonstrates the effectiveness of employing learnable shape primitives.

4.2.3 Multiple Category Point Cloud Generation

As an advanced version of TDPNet, our framework is also capable of solving a more general multiple-category problem. In this section, we evaluate all methods in such a setting, where the training set contains shapes from multiple classes. Table 4.3 shows the performance of competing methods on ModelNet and Table 4.4 shows the performance on ShapeNet. We found that our method outperforms all baselines. Particularly, our model acquires significant improvement on EMD without directly optimize it.

Moreover, compared to TDPNet, which relies on the label of each shape to retrieve correct prototype features, our method achieves better results without using any label information. This observation further demonstrates the superiority of the learnable shape primitives. Visualizations of the generated point clouds are shown in Figure 4.4 and Figure 4.6. Our method generates much more realistic and smoother point clouds (e.g., airplane cabin and chair arm) compared to baselines. Based on these observations, we conclude that the quality

Table 4.3. Performance comparison between baselines and our method on ModelNet in the multi-category setting. We report the results of each framework in the format of CD (x10^3) / EMD (x10^2). The average performance among all categories is shown in column **AVG**. Both metrics are computed on point clouds with 2048 points, and the best results are highlighted in bold.

	PSGN	FoldingNet	AtlasNet P32	TDPNet K8	L-SHAP (Ours)
Airplane	20.92 / 42.54	5.90 / 27.67	8.03 / 33.56	5.68 / 21.74	**5.56** / **14.83**
Bathtub	44.83 / 53.05	9.44 / 28.45	16.17 / 37.27	8.73 / 18.17	**7.23** / **12.99**
Bed	18.77 / 49.82	7.51 / 22.09	8.11 / 15.08	7.50 / 17.90	**6.33** / **12.98**
Chair	23.12 / 44.99	8.84 / 32.28	14.10 / 27.52	9.52 / 23.85	**7.32** / **15.74**
Desk	27.79 / 49.06	17.50 / 35.32	21.32 / 41.49	**16.61** / 27.84	16.78 / **20.52**
Dresser	54.45 / 56.65	10.38 / 26.69	17.64 / 22.37	9.74 / 20.35	**9.43** / **15.19**
Monitor	31.83 / 50.88	9.93 / 26.29	13.08 / 20.76	9.09 / 20.01	**7.91** / **15.31**
Sofa	16.59 / 50.19	7.07 / 23.92	10.10 / 16.88	7.64 / 20.30	**6.11** / **12.93**
Table	22.45 / 47.88	7.41. 25.51	9.23 / 25.74	7.03 / 18.36	**5.99** / **13.55**
Toilet	23.29 / 49.67	9.88 / 31.34	9.89 / 30.17	8.96 / 27.75	**8.04** / **16.54**
AVG.	27.14 / 49.17	9.22 / 27.78	12.38 / 26.38	8.94 / 21.74	**7.97** / **15.08**

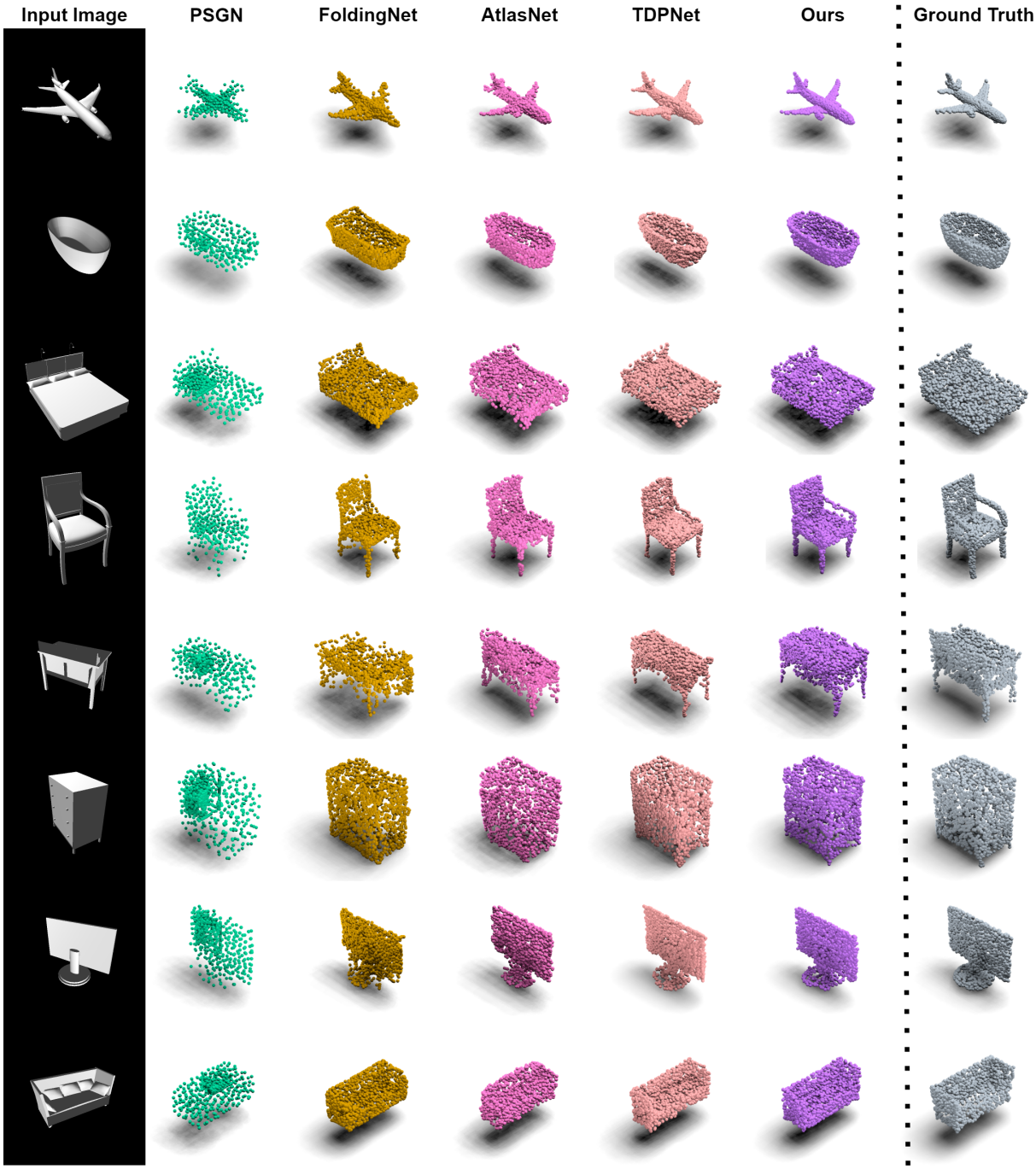

Figure 4.4. Qualitative comparison of competing methods on ModelNet. From left to right: Input image, PSGN, FoldingNet, AtlasNet P32, TDPNet K8, Our method and Ground Truth.

Table 4.4. Performance comparison between baselines and our method on ShapeNet in the multi-category setting. The results are organized in the same format as Table 4.3.

	Airplane	Chair	Car	**AVG.**
PGSN	17.85 / 36.99	24.08 / 43.96	20.83 / 48.06	21.47 / 43.01
FoldingNet	2.81 / 12.07	9.15 / 23.91	4.86 / 13.59	6.19 / 17.78
AtlasNet P32	2.81 / 13.46	8.72 / 32.69	5.10 / 12.62	6.11 / 22.07
TDPNet K8	2.51 / 11.43	8.59 / 20.01	5.41 / 12.23	6.04 / 15.57
L-SHAP (Ours)	**2.43 / 11.33**	**8.25 / 19.86**	**4.51 / 11.96**	**5.54 / 15.44**

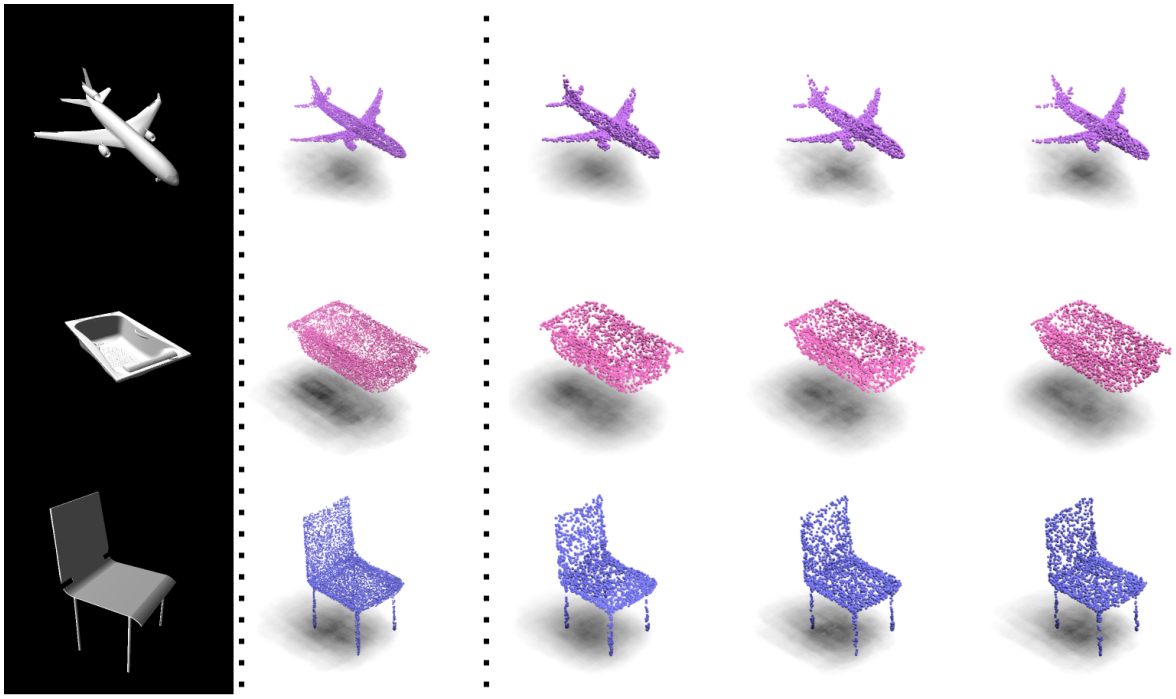

Figure 4.5. Visualization of dense generation and sampled sparse point clouds. From left to right: Input image, dense generation (10240 pts) and three sampled sparse point clouds (2048 pts).

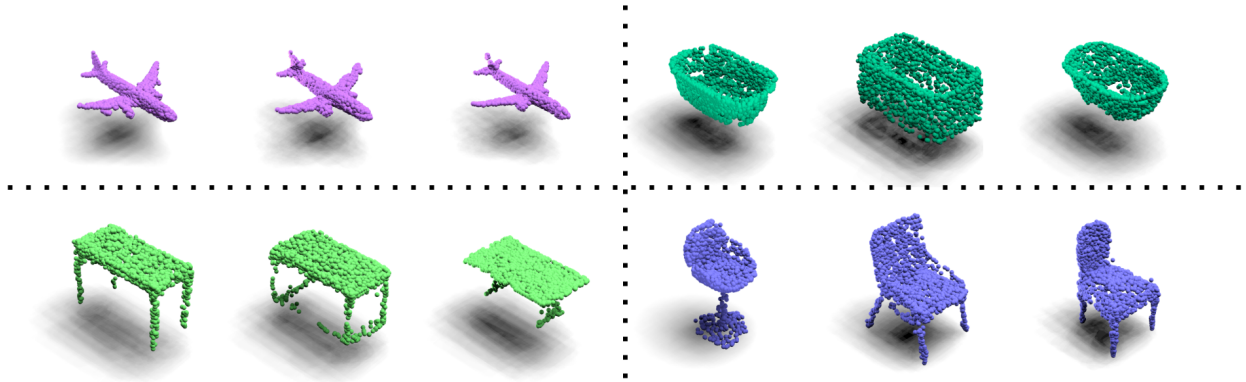

Figure 4.6. More qualitative results generated by our method. From top left to bottom right: Airplane, Bathtub, Table and Chair.

of single-view point cloud reconstruction can be significantly improved by incorporating the proposed learnable shape primitives.

4.2.4 Generating Dense Point Clouds

By incorporating continuous shape primitives, we gain the ability to generate dense point clouds. In addition, instead of using a complex model like VAE (Kingma and Welling, 2013), uniformly sampling points from the generated dense point cloud inherently offers multiple sparse point clouds. This is a valuable property because of the ambiguity in single-view 3D shape perception (Wu et al., 2018). Some examples of generated dense point cloud and a few corresponding sparse point clouds are shown in Figure 4.5. Noted the difference between the sparse point clouds, our method can generate multiple plausible point clouds from a single image.

4.3 Ablation Studies and Discussion

In this section, we conduct more experiments to exam the effect of different types of shape primitives, study the contribution of each primitive, and advise a more general high-dimensional shape primitive.

Table 4.5. Chamfer Distance (x10^3) measured on multi-category ModelNet with different "initial point set (transformation)". "2D Fix" and "3D Fix" denote fixed points uniformly sampled from unit square and unit cube, respectively.

	Airplane	Bathtub	Chair	**AVG.**
2D Square (Continuous)	5.56	**7.23**	**7.32**	7.97
2D Fix (Discrete)	**5.47**	7.42	7.45	**7.73**
3D Sphere (Continuous)	5.69	7.85	7.89	8.09
3D Gaussian (Continuous)	5.69	7.82	7.85	8.11
3D Fix (Discrete)	5.54	7.29	7.37	7.89

4.3.1 Continuous and Discrete Primitives

To investigate the influence of two proposed shape primitive transformations, we conduct ablation experiments on ModelNet with different primitive configurations. The results are reported in Table 4.5 and we observed that "2D Fix" and "3D Fix" surpass their competitors in the same dimension. Therefore, discrete shape primitives would be a good choice if dense point cloud reconstruction is not our primary goal. Interestingly, we found that increasing the dimension of the initial point sets may even hrut the performance. One potential cause of the downgraded performance is the property of the initial point set. Let's consider a 3D E^{3d}_{init} (spherical surface) and a 2D E^{2d}_{init} (unit square). E^{3d}_{init} actually contains extra "curvature" information, which might not be the desired one. Thus, maybe the neural network uses its first layer to remove such curvature information and uses the rest layers to encode useful 3D prior information. AtlasNet (Groueix et al., 2018) also observed the same phenomenon when it trying to generate a high-resolution mesh: "The output quality depends on how well the underlying surface can be represented by a sphere".

4.3.2 Contribution of Shape Primitives

We then study how shape primitives contribute to the final results and Figure 4.8 visualizes the primitives learned under the single category setting (Top) and the multiple category setting (Bottom). Although it is hard for humans to interpolate what exactly do these

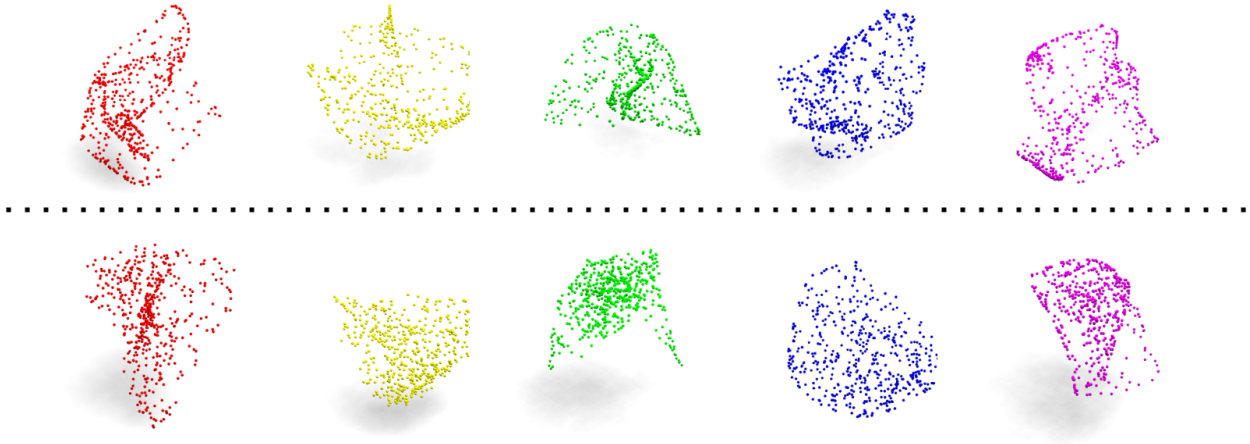

Figure 4.7. Visualizations of learned shape primitives (5 out of 8) for "airplane". TOP: primitives learned from ModelNet airplane. Bottom: primitives learned from all classes of ModelNet.

Table 4.6. Chamfer Distance (x10^3) measured on the multi-category ModelNet with different number of shape primitives K.

	Airplane	Bathtub	Chair	**AVG.**
$K = 2$	5.11	7.64	7.67	8.09
$K = 4$	5.10	7.47	7.52	8.05
$K = 8$	5.56	7.23	7.32	7.97
$K = 16$	5.52	7.19	7.27	7.92

primitives learned from the data, we observed that the multi-category shape primitives tend to be more compact and diverse comparing to the single-category version. Potentially, this might be the reason we got better performance in the ModelNet multi-category generation. Please note that, the same shape primitive always contributes to the same component in the output (e.g., cabin and tail), which indicates that they actually learned useful 3D information. Finally, we examine how the number of primitives affects the performance and report the results in Table 4.6. Notice how our approach generally improves as we increase the number of shape primitives.

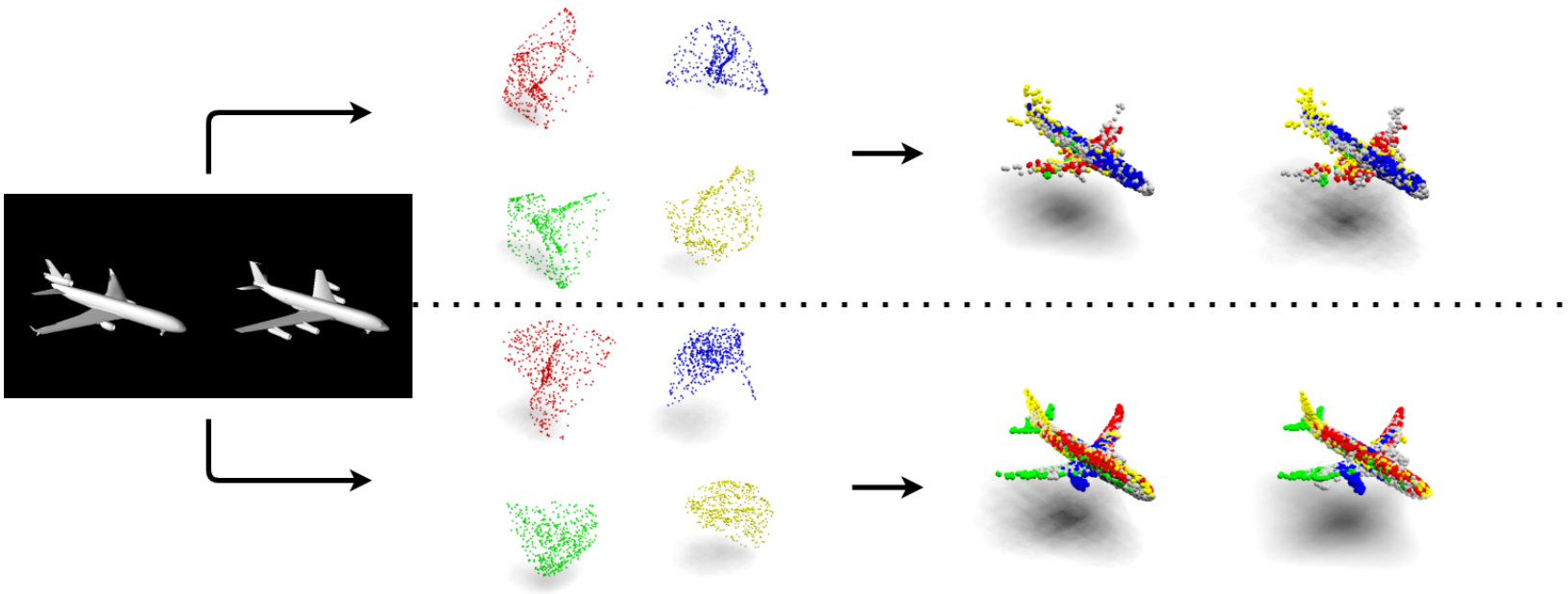

Figure 4.8. Contribution of shape primitives to the generated point cloud. Top: contribution of single-category primitives. Bottom: contribution of multi-category primitives.

Table 4.7. CD (x10^3) measured on multi-category ModelNet with shape primitives in different dimension.

	Airplane	Bathtub	Chair	AVG
$D = 3$	5.56	7.23	7.32	7.97
$D = 10$	5.40	**7.06**	**6.98**	7.84
$D = 20$	**5.14**	7.53	7.28	**7.81**

4.3.3 High-Dimensional Shape Primitives

Shape primitives are combined with image features to generate the final point clouds. Therefore, they can be conceptually in a high-dimensional space as a trade-off between performance and interpretability. Table 4.7 shows the performance of shape primitives in dimension 3, 10, and 20. In opposition to the initial point sets, we achieve better generation performance by increasing the dimensionality. This observation suggests that higher-dimensional primitives encode more complex 3D information and thus should be adopted for complex shapes. Nevertheless, it's noteworthy that this improvement diminishes for higher-dimensional space.

4.3.4 Discussion

In this chapter, we introduced a 3D-aware framework that generates point clouds from an image through shape primitives. Instead of manually defining the structures of shape primitives, our shape primitives are learned from data. Extensive experiments with different configurations on different datasets demonstrate that our model outperforms other state-of-the-art methods regarding quantitative metrics and visual quality. Moreover, we empirically study the effect of high-dimensional primitives and different primitive transformations. Overall, these observations demonstrate the effectiveness of our method and investigating the impact of different parameterization of shape primitives could be an interesting research direction in the future.

CHAPTER 5

GENERATING POINT CLOUD FROM SINGLE IMAGE IN THE FEW SHOT SCENARIO [1]

5.1 Approach

Previous point cloud generation frameworks (Fan et al., 2017; Yang et al., 2018; Groueix et al., 2018; Lin et al., 2021) generally require heavy supervisions and thus less applicable by considering the size and variety of point cloud datasets. As shown in Figure 5.1, abundant training pairs are practically inaccessible, thus successfully generating point cloud in a few-shot configuration is amenable to many real-world applications.

Interestinlgy, humans are able to solve this 3D reconstruction problem seamlessly. The main reason is that people can accumulate and utilize prior knowledge about 3D shapes, even from limited demonstrations. Based on these observations and recent advances of few-shot learning (Vinyals et al., 2016; Snell et al., 2017), we suggest that single-view point cloud reconstruction is of particular interest under the few-shot learning. To be more concrete, we argue that point clouds can be reconstructed from a single image by considering a class-specific prototype extracted from limited support samples, and class-agnostic shape primitives learned from the training data.

In this chapter, we propose a novel few-shot single-view point cloud generation network accompanied with a new training strategy, namely "episodic training", to assist learning from scarce data. We first assume that the 3D shape prior can be divided into two parts: 1) class-specific and 2) class-agnostic shape priors. The intuition behind this assumption is straightforward: a 3D shape generally consists of class-specific shape elements (e.g., wings of airplanes) and class-agnostic shape elements (e.g., legs of chairs and tables). Specifically, we

[1]This chapter contains material previously published as: Yu Lin, Jinghui Guo, Yang Gao, Yifan Li, Zhuoyi Wang, and Latifur Khan. "Generating Point Cloud from Single Image in The Few Shot Scenario". In *Proceedings of the 29th ACM International Conference on Multimedia*, pp. 2834-2842. 2021

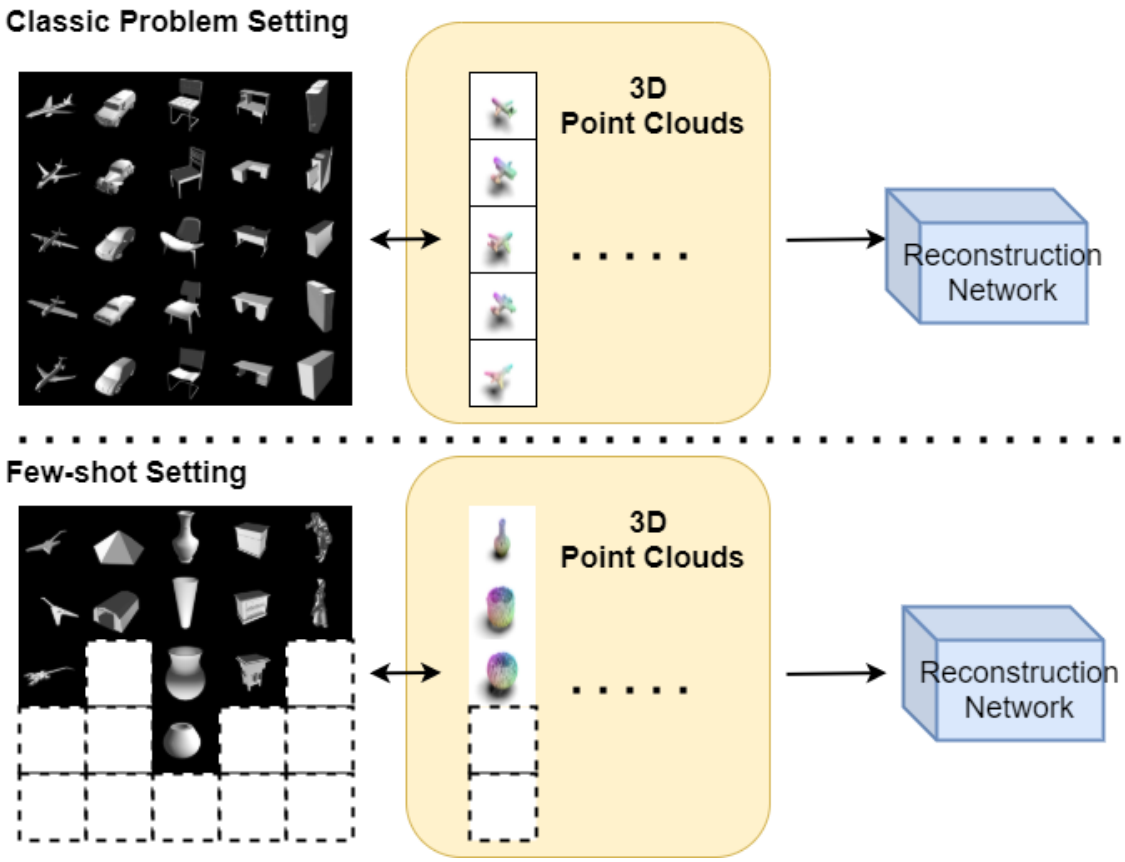

Figure 5.1. A high-level comparison between the classical problem setting (abundant training pairs) and the few-shot setting (insufficient training instances).

model the class-specific shape prior as a latent vector computed from limited point clouds of same class. Inspired by AtlasNet and its variants(Groueix et al., 2018; Deprelle et al., 2019), we employ a collection of shape primitives, learned from the training data, to represent class-agnostic shape priors. We then fuse the information from incoming images with both shape priors to guide the point cloud generation process.

When it comes to novel classes, our model computes their class-specific shape priors "at runtime" and utilizes the trained shape primitives, as class-agnostic shape priors, to reconstruct the point clouds without additional retraining. We show that this boost reconstruction performance significantly over state-of-the-art single-view point cloud reconstruction methods. Comparing with conventional episodic training in the few-shot classification/segmentation,

our episodes are created in a "1-way-k-shot" fashion. We additionally introduce a novel "intra-support" training procedure for each episode. Namely, the reconstruction loss is computed on both the query and support sets. We empirically demonstrate that this operation would benefit point cloud reconstructions, especially when the data is limited.

5.1.1 Framework Overview

Problem Setting

Let $D = \bigcup_{c \in C} \{I_i^{(c)}, S_i^{(c)}\}_{i=1}^{N_c}$ be a collection of image-point-cloud pairs, where C is the set of all classes, N_c denotes the number of image-point-cloud pairs belongs to class c, $I_i^{(c)}$ and $S_i^{(c)}$ indicate the i-th image and point cloud in class c, respectively. A point cloud S is a set of points $\{p_j\}_{j=1}^{M}$, where M means its cardinality and p_i is a point in the 3D Euclidean space with coordinate (x_j, y_j, z_j). Follow the same protocol of previous chapters, we choose $M = 2048$ in our experiments as it's sufficient to preserve the major structure of a given 3D object and requires reasonable computational resource (Awad and Khan, 2007; Chang et al., 2015).

Our objective is to use D to train a model that consumes a view of 3D object and a set of support point clouds to generate a point cloud of that 3D object. Following the convention of FSL (Wallace and Hariharan, 2019), we split D into two disjoint sets D_{base} and D_{novel}, C_{base} and C_{novel} are the classes in each set. The network are trained on D_{base} and tested on D_{novel}. Please note that $C_{base} \cup C_{novel} = C$ and $C_{base} \cap C_{novel} = \emptyset$. We further assume that D_{base} has abundant image-point-cloud pairs, whereas the number of paired data in D_{novel} is much smaller.

During the test phase, the model encounters image $I^{(c')}$ from novel classes $c' \in C_{novel}$ and builds class-specific shape prior from a minor support set. Our goal is to minimize the distance between the synthesizations and corresponding ground truths measured by the following famous metrics: Chamfer Distance (CD) and Earth Mover Distance (EMD). Recall

that their formulas are presented in Equation 3.2 and Equation 3.3, respectively. EMD favors the shapes close to the "mean-shape" of the given category (Fan et al., 2017). In contrast, CD tends to cover all components while leading to a splashy shape that blurs the object's geometric structure.

Workflow

We employ a conventional encoder-decoder architecture with three different branches. As Figure 5.2 presented, our framework contains four major components: a point cloud encoder Enc_{pc}, a point cloud decoder Dec_{pc}, an image encoder Enc_{img}, and a learnable shape primitive (a transformable complex point distribution).

The image encoder Enc_{img}, a VGG-16 pretrained model in this paper, takes a 2D image as input and produces an informative latent representation. Following the "1-way-K-shot" FSL configuration, we require K support point clouds $\{S_i\}_{i=1}^{K}$ to compute the class-specific shape prior vector ζ_{cs}. Noted that we omitted the indicator of class, (c), because of the 1-way setting and notation simplicity. We first use a pretrained point cloud encoder Enc_{pc}, e.g., PointNet (Qi et al., 2017) or DGCNN (Wang et al., 2019), to compute the embedding of support point clouds, $\{e_i\}^K = \{Enc_{pc}(S_i)\}^K$. We then apply a simple element-wise average operation on these embeddings to get a "class-specific" shape prior vector. It's worth highlighting that such a shape prior is valuable in the few-shot reconstruction scenario because it preserves the 3D shape information from scarce novel class objects.

Nevertheless, a class-specific shape prior is insufficient for a realistic reconstruction. A good generation framework should take both class-specific and class-agnostic shape priors into consideration. Several manifold-based frameworks (Groueix et al., 2018; Yang et al., 2018; Deprelle et al., 2019) model the shape information by transforming a simple distribution to a complex and semantic meaningful distribution. Follow the same idea, we use the same technique previously proposed in L-SHAP (see Chapter 4) and model the class-agnostic shape

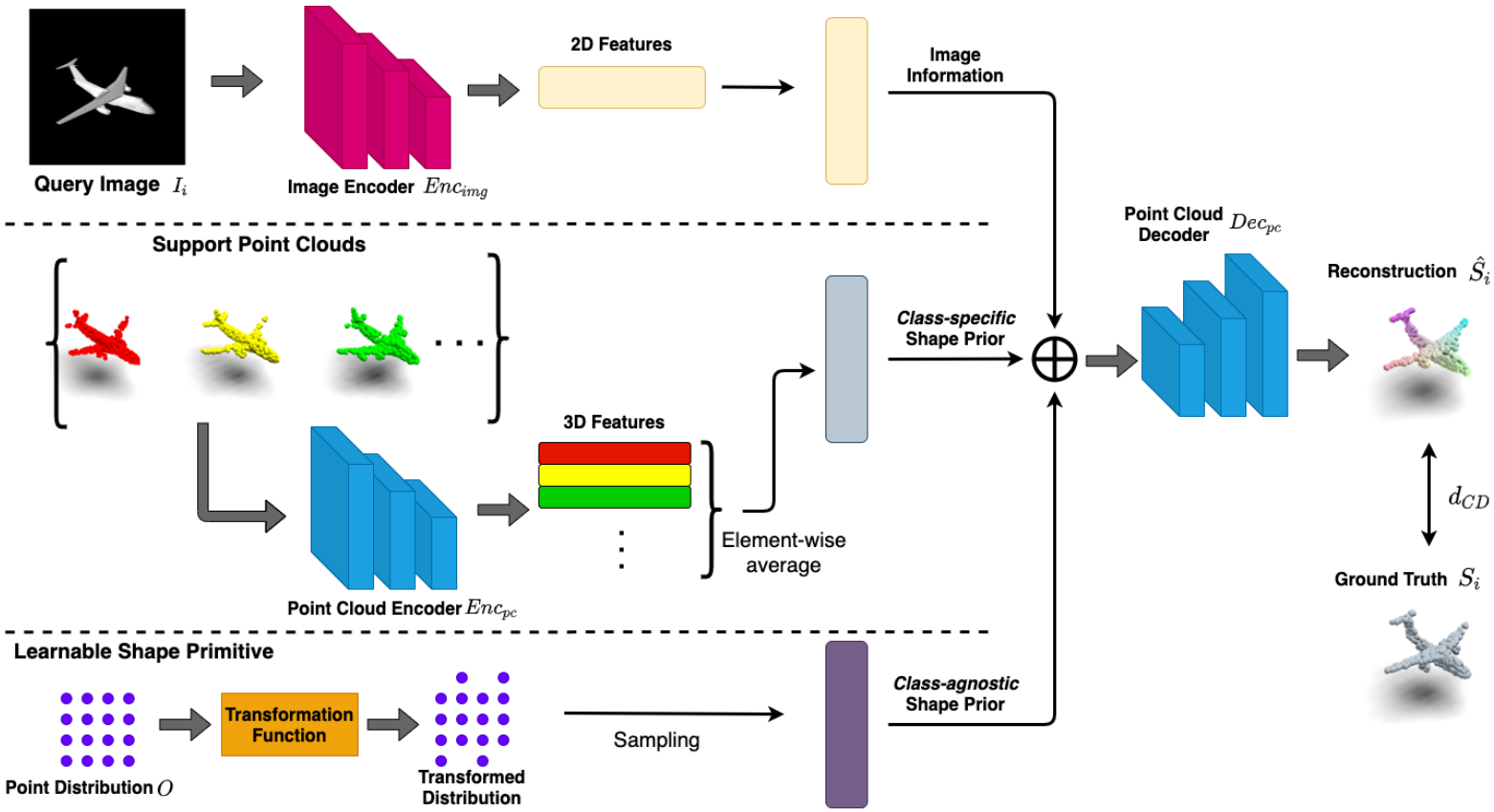

Figure 5.2. Overview of our framework. For a given query image I_i that belongs to class c' (c' is airplane in this figure), we first compute its latent representation through an image encoder Enc_{img}. In the middle branch, we compute a *class-specific* shape prior vector by averaging the 3D features of support point clouds. The *class-agnostic* shape prior is presented by a shape primitive, which is a transformed complex point distribution. Finally, we jointly decode the latent image representation, "class-specific" shape vector and "class-agnostic" shape primitive onto the target point cloud $\hat{S}$ with Dec_{pc}. Chamfer distance between $\hat{S}_i$ and corresponding ground truth S_i is computed during the training.

prior by a complex point distribution. More precisely, we represent the shape prior by a pair of initial point distribution O and a transformation function $\psi(\cdot)$ that maps O to a more complex distribution. To the opposite of class-specific shape prior that is computed from a set of support point clouds, the learnable shape primitive learns the common geometry information across classes during the training.

Finally, a point cloud decoder, Dec_{pc}, consumes the image features, class-specific shape priors and class-agnostic shape priors to generate a reconstructed 3D shape in the form of a point cloud, $\hat{S}_i$:

$$\hat{S}_i = Dec_{pc}((Enc_{img}(I_i) \oplus V_{cs}) \odot \psi(O)) \tag{5.1}$$

where $\oplus$ denotes the concatenation and $\odot$ means the point endowment operation. We will thoroughly discuss this operation in Section 5.1.2. Conceptually, the whole network can be trained using either Eq 3.2 or 3.3 between reconstrcuted 3D shape $\hat{S}$ and corresponding ground truth S.

This triple-branch architecture enjoys several advantages: (1) Retraining is not required in the test phase because class-specific shape prior is computed from the support set. (2) There is no schematic difference between the training (base classes) and the testing (novel classes) (Masud et al., 2015; Michalkiewicz et al., 2020). Class-specific shape priors are directly computed from the support set. (3) The performance of generation are significantly improved by explicitly considering both class-specific and class-agnostic shape priors.

5.1.2 Class-specific and Class-agnostic 3D Shape Priors

In the section, we will introduce class-specific and class-agnostic shape priors in detail.

Class-specific Shape Priors

An RGB image solely contains deficient information for a complex 3D shape, especially in an FSL scenario. To tackle this issue, 3D priors are introduced to compensate for such

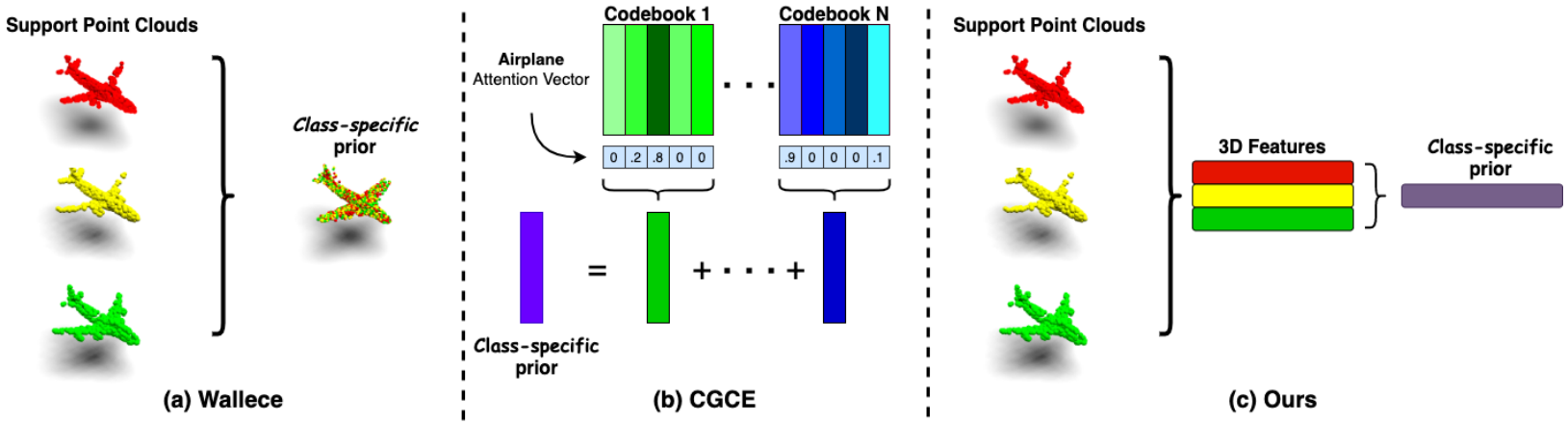

Figure 5.3. Comparison of three different "class-specific" shape priors. (a) Due to the irregularity of point clouds, all support point clouds are combined into a dense one in the original Euclidean space. (b) CGCE defines N learnable codebooks and compute the attentional sum of them for each class. (c) The shape prior is computed by averaging the hidden features of support point clouds. By separating two types of shape priors, our method offers a more meaningful shape prior and eliminates finetuning in the test phase.

information loss. Preceding works (Wallace and Hariharan, 2019) show that a single "mean shape" of several complex 3D shapes is a good approximation of the 3D shape prior. However, this "mean shape" approach is infeasible in the point cloud domain because of the irregularity of point cloud. Unlike voxel that a common template can be used to describe a shape (e.g., a 64^3 voxel grids), there is no point correspondence across different point clouds. Alternatively, CGCE (Michalkiewicz et al., 2020) learns a few codebooks and build class-specific shape prior via a SparseMax (Martins and Astudillo, 2016) attention layer. Disadvantages of this approach are obvious: 1) the proposed codebooks lack interpretability, and 2) finetuning on novel classes is mandatory, which is unacceptable in many applications. To mitigate these issues, we propose to build the shape prior in a latent space and Figure 5.3 provides the high-level comparison among these methods. It's worth noting that an"mean-shape" prior of point cloud (Wallace and Hariharan, 2019) in the original 3D Euclidean space is not applicable due to the non-Euclidean property, thus we compute the element-wise average of E_{pc} as the class-specific shape prior, which is also known as class prototype (Snell et al., 2017). Let $\{S_i\}^K$ be the set of support entities, the class-specific shape prior vector V_{cs} is formulated as:

$$V_{cs} = \frac{1}{K}\sum_{i=1}^{K} e_i = \frac{1}{K}\sum_{i=1}^{K} Enc_{pc}(S_i) \tag{5.2}$$

Class-agnostic Shape Priors

Before diving into the class-agnostic shape prior, let's first revisit two popular point cloud decoder architectures in Figure 5.4. Most previous approaches (Abrol and Khan, 2010a; Fan et al., 2017; Mandikal et al., 2018; Achlioptas et al., 2018; Chibane et al., 2020; Liu et al., 2019) and two FSL methods (Wallace and Hariharan, 2019; Michalkiewicz et al., 2020) follow the first schema, although the implementation of the decoder varies (e.g., MLP and GCN). The major disadvantage of this approach is that the generated point cloud has a certain amount of points, which suffers when more points are required. On the other hand, the point

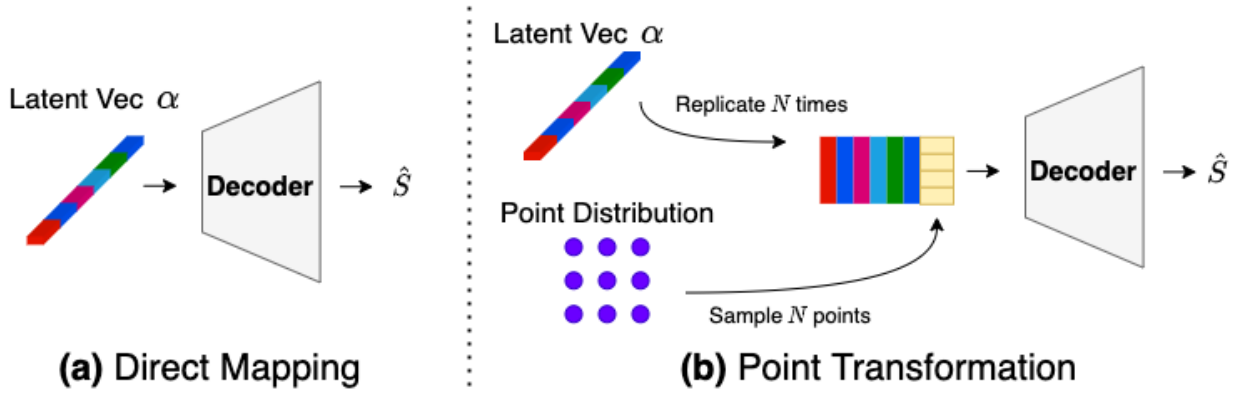

Figure 5.4. Comparison of two decoding schema. Assume target shape has N points and α is a latent vector. (a) directly maps α to a point cloud $\hat{S}$. (b) samples N points from a point distribution, endows each point with α (concatenate its coordinates and α), and transforms the endowed point onto target point cloud $\hat{S}$.

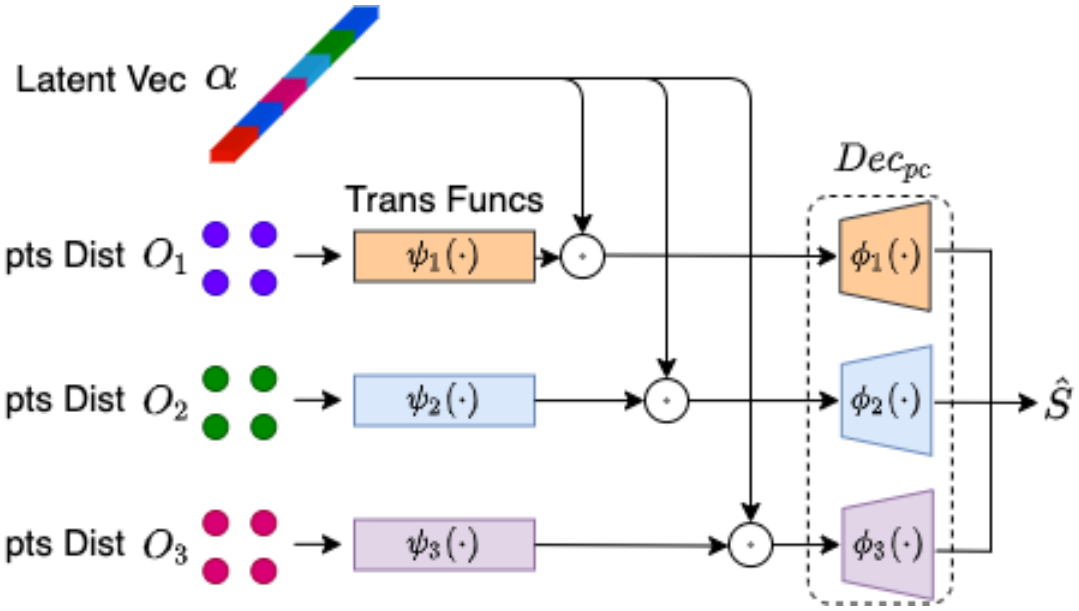

Figure 5.5. We employ M learnable primitives ($M = 3$). Dec_{pc} contains M sub-decoders $\phi_m(\cdot)$ and all outputs of them are collected onto $\hat{S}$.

transformation methods (Yang et al., 2018; Groueix et al., 2018; Deprelle et al., 2019) are able to generate as many points as demanded by sampling more points from the distribution.

Our decoder follows the second schema because of its flexibility and superior performance. Moreover, we modified its architecture to store *class-agnostic* shape priors. Inspired by

FoldingNet (Yang et al., 2018) and AtlasNet (Groueix et al., 2018), we propose to encode the shape prior by a set of learnable shape primitives (see Figure 5.5), each primitive is a complex point distribution transformed from a simple one: $\{(x, y)|x \sim U(0, 1), y \sim U(0, 1)\}$. With this configuration, each shape primitive stores part of the "class-agnostic" shape information during the training and democratically contributes to the final output. Formally, the generation process can be written as:

$$\hat{S}_i = Dec_{pc}(\alpha) = \bigcup_{m=1}^{M} \bigcup_{p \in O_m} \phi_m(\psi_m(p) \oplus \alpha) \tag{5.3}$$

where α is the concatenation of image feature and *class-specific* shape vector. $\psi_m(\cdot)$ and $\phi_m(\cdot)$ denote the transformation function and decode function for point distribution O_m, respectively.

5.1.3 Intra-Support Episodic Training

We start by showing how to build an episode and then discuss the benefits provided by this training strategy. The episodes are built in a "1-way-k-shot" fashion (Fei-Fei et al., 2006). We uniformly sample K and Q image-point-cloud pairs, without replacement, as the support set and query set, respectively.

We choose episodic training due to the following reasons: (1) It guarantees the consistency between the training phase and testing phase, which encourages the proposed framework learning to extract discriminative information in a few-shot setting. Noted that previous methods (Wallace and Hariharan, 2019; Michalkiewicz et al., 2020) assume plentiful training data are available and compute shape prior from the whole training set, which introduce a discrepancy between training and testing environments. (2) Since the episodes are randomly sampled from data with significantly different configuration of support and query sets (e.g., data forming these sets vary from iteration to iteration), it requires the network to capture the underlying shape concepts that are common among different episodes and different classes.

Algorithm 4: Episodic Training

input : A set of image-point-cloud pairs D_{base}

```
for Number of training epochs do
    for Number of training episodes do
        Sample a training class from C_base
        C ← RANDOMSAMPLE(C_base);
        Sample K instances from class C
        Sup ← RANDOMSAMPLE(D_base, C, K);
        Sample Q instances from class C
        Que ← RANDOMSAMPLE(D_base, C, Q);
        Generate Ŝ for K + Q images with Equation 5.1;
        Compute reconstruction loss with Equation 5.4;
        Backward and update the network ;
    end
end
```

We further introduce a novel intra-support augmentation to boost the training performance by reconstructing point clouds in the support set. To be concrete, instead of computing the reconstruction loss merely on the query instances, we expand the query set to includes images from the support set, results in $K + Q$ training pairs. The insight behind this operation is straightforward yet intuitive. We should be able to reconstruct a complete 3D point cloud from its own 3D feature and 2D projections by using our framework. This operation is highly useful, especially when the support set and query set are small. Let Sup and Que be the support and query set respectively, $d(\cdot, \cdot)$ denotes the distance function (Equation 3.2 or Equation 3.3) and λ be the weight factor, our final objective is formulated as:

$$L = L_{Que} + \lambda L_{Sup} = \sum_{Que} d(\hat{S}, S) + \lambda \sum_{Sup} d(\hat{S}, S) \tag{5.4}$$

The overall training procedure is illustrated in Algorithm 4.

5.2 Evaluation

The performance of the proposed method is evaluated both quantitatively and qualitatively. We provide the results for both base classes and novel classes simultaneously. We also report the model performance with different sizes of the support set. Moreover, ablation studies are performed to analyze the contribution of each individual module to the model performance.

5.2.1 Experiment Setting

Two datasets, a subset of ModelNet (Wu et al., 2015) and a subset of ShapeNet (Chang et al., 2015), are adopted for evaluations in our experiments. For each dataset, we selected five categories as novel classes, C_{novel}, and the rest categories are considered as base classes, C_{base}. Both D_{base} and D_{novel} are further split into train/test split in a 80/20 pattern. For each 3D shape in the datasets, we render 12 views of it based on the *Blinn-Phong* shading formula with black background. For each object, a single view image is uniformly sampled from its 12 2D-projections to form the training data. As suggested in previous literature (Abrol and Khan, 2010b; Fan et al., 2017; Achlioptas et al., 2018), we evaluate the performance of our model with Chamfer Distance (CD) and Earth Mover Distance (EMD).

We adapt a pretrained VGG-16 as the image encoder, where all layers except the last three convolutional layers are freezed during training. The point cloud encoder has same architecture as PointNet (Qi et al., 2017) and is trained from scratch. Our decoder contains four learnable shape primitives. For each shape primitive, its transformation function $\psi(\cdot)$ is implemented by a 2-128-128-3 MLP with ReLU non-linearities on the first two layers and tanh on the last output layer. Each decode function $\phi(\cdot)$ is implemented again by a 1539-769-384-3 MLP that uses ReLU activation in the first three layer and tanh in the last layer. Both transformation function and decode function apply batch normalization in all layers except the last one.

Before the training, we align the input point clouds to a common ground plane and normalized all points into a unit ball. Data augmentation strategies like random rotation and jitter are applied during the training. All the RGB images are center-cropped and resize to 224×224. To train our model, we use an ADAM optimizer with an initial learning rate of 10^{-3} and $\beta = (0.9, 0.999)$. Each training epoch contains 100 episodes. A step learning rate scheduler with $\gamma = 0.5$ is employed to decay the learning rate every 300 epochs. The model is trained on a RTX Quardo 8000 GPU with 1000 epochs.

5.2.2 Baselines

We compare against several state-of-the-art (SOTA) baselines in the single-view point cloud reconstruction domain. Specifically, we consider the zero-shot (ZS) and fine-tune (FT) variants of three SOTA fully supervised single-view point cloud reconstruction methods and two FSL 3D reconstruction methods.

- **PSGN** (Fully Supervised) (Fan et al., 2017): This is the first learning-based single view point cloud reconstruction framework. It uses a hybrid decoder and is trained with a MoN loss.

- **AtlasNet** (Fully Supervised) (Groueix et al., 2018): It's a pioneer approach the embraces the manifold assumption. It deforms multiple feature-endowed patches onto the target's surface.

- **LMNet** (Fully Supervised) (Mandikal et al., 2018): This approach applies the idea of multi-modal fusion. It first train a point cloud auto-encoder and approximate the trained point cloud encoder an image encoder.

- **Wallece** (FSL) (Wallace and Hariharan, 2019): It's the first few-shot 3D reconstruction framework. A shape prior (template) in the original 3D Euclidean space is obtained from the support set and image features are used to refine the template.

- **CGCE** (FSL) (Michalkiewicz et al., 2020): This framework introduces a hierarchical model of shape prior. The shape priors of novel classes are obtained by finetuning the network on support instances.

To avoid the discrepancy caused by different image encoders, all image encoders in baselines and our method are replaced by a pretrained VGG-16. Note that all layers except the last three convolutional layers in VGG-16 are freezed during training. Moreover, since both FSL methods target on voxel representation and use direct mapping schema, we switch their voxel decoder with a LMNet point cloud decoder.

The ZS-baselines are solely trained on the dataset D_{base}. These methods provide sorts of performance lower bound on the novel classes since the model observes no data from novel classes. On the other hand, the FT-baselines are pre-trained on the base classes D_{base} and fine-tuned on a small support set. Finetuning is a famous meta-learning strategy to reduce the domain discrepancy, and we would like to study its performance in a few-shot scenario. Compared with conventional funetine methods, FT-baselines are finetuned on an episodic way. As we discussed in previous sections, each episode contains one test instance and a few randomly sampled support instances. FT-baselines are finetuned on the support set and evaluated on the query set. For a fair comparison, we choose $K = 32$ for the purpose of finetuning, which is consistent with the largest number of shots in the few-shot setting.

5.2.3 Novel Classes Reconstruction

To evaluate the effectiveness of the proposed method, we first conduct the experiment with a relatively large support set ($K = 32$), since the performance of ZS-baselines and FT-baselines are nearly indistinguishable if the support set is too small. Table 5.1 and Tab 5.2 present the Chamfer Distance (CD) and Earth Mover Distance (EMD) between the generated point clouds and ground truth, respectively. It demonstrates that the proposed method consistently achieves better CD in most categories for the novel classes reconstruction. It is also observed

Table 5.1. Few-shot single-view reconstruction (32-shots per category) for both datasets. We report the Chamfer Distance (CD) of each framework and the values are multiplied by 10^2 for better visualization. The average performance among all categories (per dataset) is shown in row AVG. Metric is computed on 2048 points and best results are bolded.

	Category	**PSGN**		**LMNet**		**AtlasNet**		***Wallace***	**CGCE**	**Ours**
		ZS	FT	ZS	FT	ZS	FT			
ModelNet	Bowl	15.06	14.69	16.53	15.99	11.33	11.17	12.90	2.83	**1.98**
	Cup	10.81	10.63	9.72	10.21	7.27	7.44	16.19	3.10	**2.65**
	Door	4.49	4.17	10.94	9.68	16.68	10.74	24.29	3.67	**2.41**
	Keyboard	4.13	3.38	8.84	9.61	10.77	10.34	5.56	3.24	**3.18**
	Laptop	5.13	4.99	7.88	6.62	7.09	6.74	11.85	**1.13**	2.01
	AVG.	7.92	7.57	10.78	10.42	10.63	9.29	14.16	2.79	**2.45**
ShapeNet	Bowl	12.41	12.79	5.28	9.19	10.83	10.83	8.01	**2.67**	4.62
	Cellphone	3.46	2.09	8.89	8.70	26.42	26.11	12.45	1.73	**1.16**
	Jar	13.23	12.25	6.29	7.07	16.80	16.59	14.21	5.81	**5.52**
	Monitor	3.52	3.18	3.61	6.87	19.65	19.36	8.20	3.96	**2.49**
	Mug	15.08	14.01	3.32	6.96	9.88	9.64	11.54	3.43	**2.21**
	AVG.	7.38	6.70	5.26	7.39	18.77	18.52	10.65	3.91	**3.13**

Table 5.2. Few-shot single-view reconstruction (32-shots per category) for both datasets. We report the Earth Mover Distance (EMD) of each framework and the values are multiplied by 10^2 for better visualization. The average performance among all categories (per dataset) is shown in row AVG. Metric is computed on 2048 points and best results are bolded.

	Category	**PSGN**		**LMNet**		**AtlasNet**		***Wallace***	**CGCE**	**Ours**
		ZS	FT	ZS	FT	ZS	FT			
ModelNet	Bowl	63.67	52.11	8.93	8.51	36.23	18.37	8.96	2.95	**1.08**
	Cup	55.87	33.40	7.46	7.84	35.03	19.61	9.79	4.17	**1.42**
	Door	53.50	38.11	10.12	10.10	38.61	16.53	14.26	5.84	**2.32**
	Keyboard	46.41	36.62	7.44	7.69	36.24	24.16	7.92	5.36	**2.75**
	Laptop	45.96	38.65	5.42	4.53	29.95	26.11	7.15	1.99	**1.07**
	AVG.	53.08	39.78	7.87	7.73	35.21	20.96	9.62	4.06	**1.73**
ShapeNet	Bowl	63.14	46.84	7.91	5.61	26.01	21.75	5.08	3.03	**1.10**
	Cellphone	51.03	30.71	6.49	8.55	44.30	16.54	9.70	4.00	**2.57**
	Jar	54.62	40.52	7.42	6.09	33.04	20.33	8.94	6.15	**1.93**
	Monitor	50.09	27.94	6.95	6.69	33.47	16.48	6.12	4.44	**1.12**
	Mug	56.99	49.99	7.33	4.55	27.05	15.52	6.31	3.53	**1.79**
	AVG.	52.76	34.45	7.08	6.62	34.23	17.72	7.37	4.62	**1.62**

that the performance improvement offered by finetuning is limited in the few-shot setting. For example, the average CD of ModelNet only reduced from 7.92 to 7.57 (10.78 to 10.42) by finetuning the pretrained PSGN (LMNet). It shows that, on the other hand, our approach consistently wins in a large margin.

Moreover, compared with to CGCE and our method, Wallace's framework does not achieve satisfying performance in these experiments, which shows that building class-specific shape prior in the original space is not advisable in the point cloud domain. We also observed that although both CGCE and our method build the shape priors in a latent space, our method requires no finetuning in during the test phase and is able to achieve better scores in both metrics. This phenomenon further demonstrates the effectiveness of separating class-specific and class-agnostic shape priors.

We then study how the performance of our method and CGCE evolve as the support set becomes bigger in Figure 5.6 and Figure 5.7. We observed that the average CD between the ground truths and generated point clouds is large when the size of the support set is small, especially when $K = 1$, which is known as the challenging one-shot problem. Notice how our approach generally improves as we increase the number of support point clouds. We also found that there is a sharp performance boost when the number of support point clouds is increased from 1 to 4, or 4 to 8. However, the benefits offered by extra support data vanished after 8-shots, indicated by the almost flatten lines after $K = 8$. Additionally, it is noteworthy that CGCE may not work well in a low-shot scenario. For example, the one-shot CGCE got 75.83 (88.72) Chamfer Distance on bowl (door) and 42.34 (52.93) Earth Mover Distance on cup (door), which exceed the scope of the line plots.

In contrast, our method conducts a good generation in the low-shot scenario. Additionally, an example of qualitative result is presented in the top of Figure 5.8. It is observed that our method can generate clear and realistic point clouds from novel classes in a few shot setting (number of shots $k = 16$).

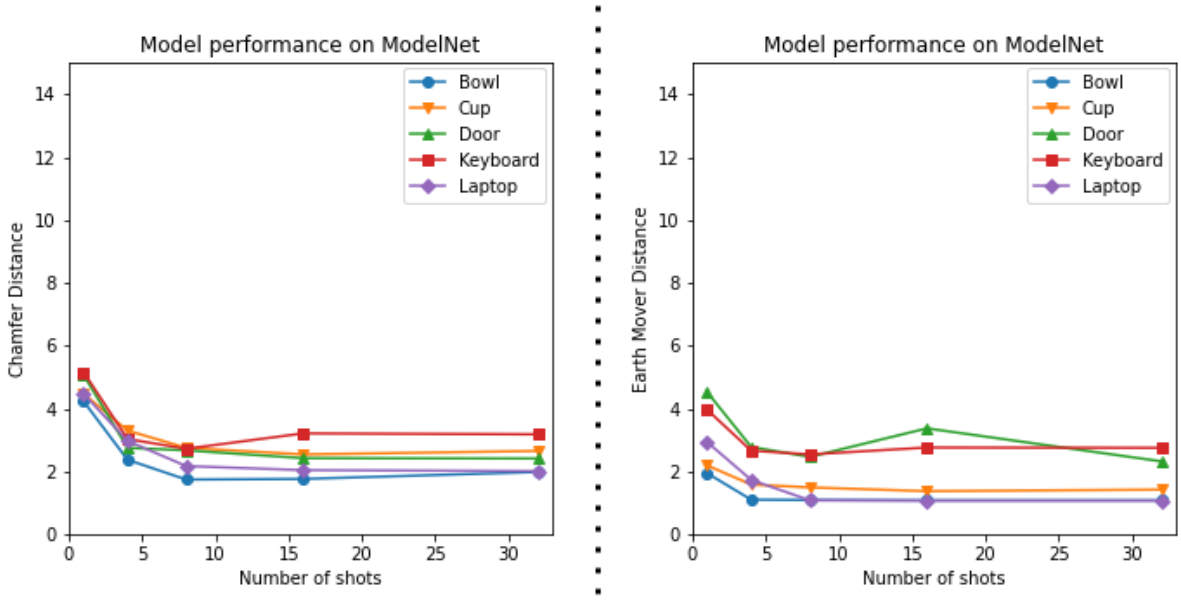

Figure 5.6. Model Performance of FPSG under different values of shots ($K = 1, 4, 8, 16, 32$) measured on ModelNet. Left plot is for Chamfer Distance and right plot is for Earth Mover Distance.

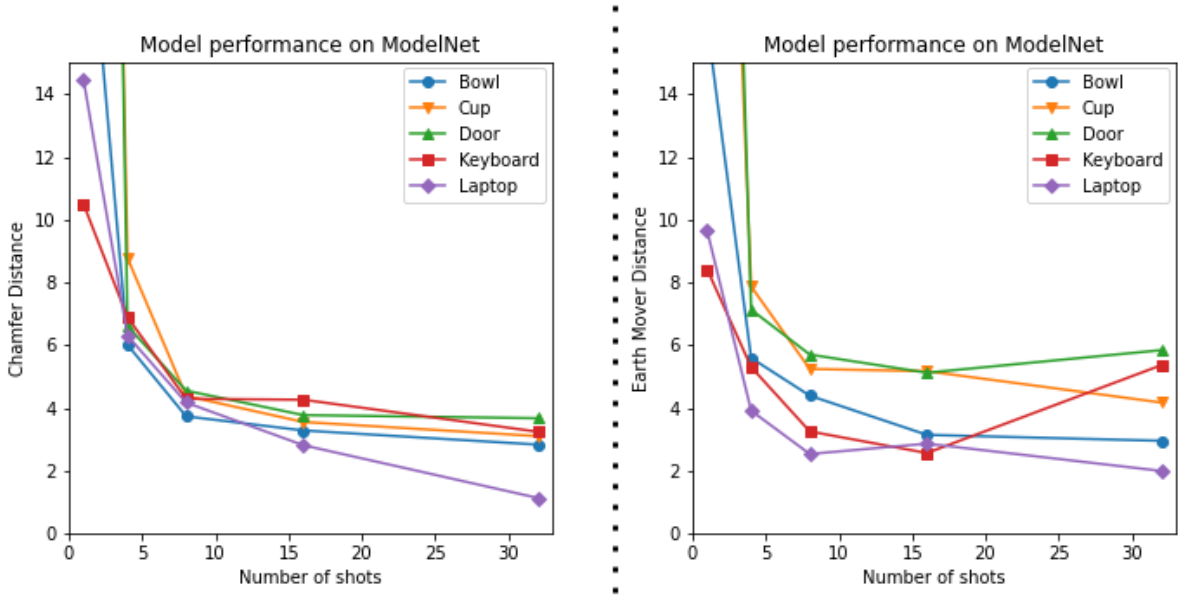

Figure 5.7. Model Performance of CGCE under different values of shots ($K = 1, 4, 8, 16, 32$) measured on ModelNet. Left plot is for Chamfer Distance and right plot is for Earth Mover Distance. Missing values are explained in the context.

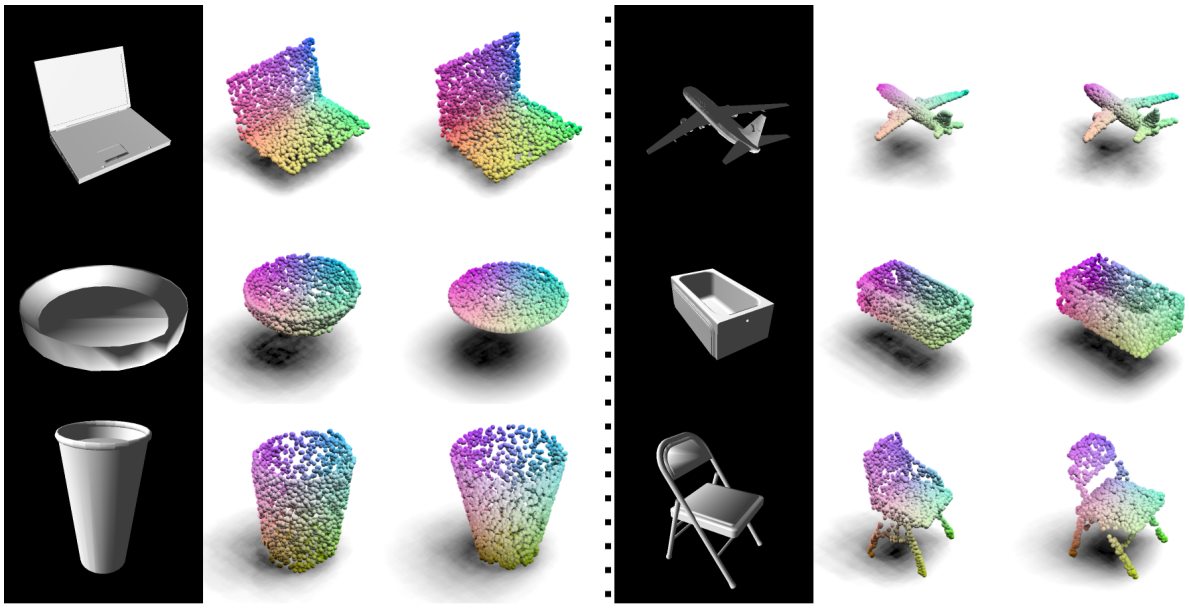

Figure 5.8. Examples of generated point clouds with number of shots $k = 16$. Novel classes are in the left (laptop, bowl and cup) and base classes are in the right (airplane, bathtub and chair). For each side, from left to right are: input image, generated point cloud, and ground truth.

Based on these experimental performances, we conclude that the quality of generated point cloud, in the few-shot scenario, can be significantly improved by using the proposed method.

5.2.4 Base Classes Reconstruction

Although our method focuses on the few-shot reconstruction problem, it would be ideal if we can achieve reasonable reconstruction on the training classes comparing to the heavily supervised methods. In this experiment, we ignore the novel classes C_{novel} and focus on the base classes C_{base}. Specifically, we split the base classes into two subsets, $C_{base-train}$ and $C_{base-test}$, in a classical way (80/20 split). We use the same episodic training schema to train the network on $C_{base-train}$ and tested it on the $C_{base-test}$.

Since the training data is sufficient for the base classes, we reduce the number of shot K from 32 to 16 in this experiment. Tab 5.3 shows the Chamfer Distance measured on a

Table 5.3. CD measured on a subset of ModelNet $C_{base-test}$. Number of shots $k = 16$ and the CDs are scaled by 10^2. Best results are bolded and the seconds are underlined.

	PSGN	LMNet	AtlasNet	Wallace	CGCE	Ours
Bathtub	4.48	4.34	**1.62**	5.59	2.53	<u>1.96</u>
Chair	2.31	2.64	**1.41**	10.43	3.15	<u>1.70</u>
Desk	2.78	3.73	**2.13**	6.46	3.36	<u>2.51</u>
Dressor	5.45	4.48	2.76	14.71	**2.47**	<u>2.66</u>
Sofa	1.66	2.52	**1.01**	3.73	<u>1.41</u>	**1.01**

Table 5.4. Ablation Study on ModelNet. a check mark means the corresponding component is activated. Number of shots $k = 32$ and the CDs are scaled by 10^2.

	intra-sup	V_{cs}	V_{ca}	Bowl	Cup	**AVG.**
V1				17.71	12.13	11.18
V2	✓			17.69	12.43	11.55
V3		✓		-	-	-
V4			✓	10.14	7.01	8.98
V5	✓	✓		3.13	4.10	3.97
V6		✓	✓	-	-	-
V7	✓		✓	10.56	6.99	9.02
V8	✓	✓	✓	**1.98**	**2.65**	**2.45**

subset of the ModelNet training (base) classes. It's worth noting that the CD scores of the training classes is much smaller comparing to the novel classes, which proved the effectiveness of the three fully supervised methods. Although we cannot achieve the same performance as AtlasNet does, our method still obtain reasonable scores and outperformed other baselines. Some qualitative examples of base classes are presented on the right of Figure 5.8. We discover that our method is able to generate sharp-looking point clouds of base classes.

5.3 Ablation Studies and Discussion

We further conduct an ablation study on the proposed method to inspect the contribution of each component.

5.3.1 Contribution of Components

Specifically, we evaluate the model performance on the same dataset by alternatively activating network components: intra-support training, class-specific and class-agnostic shape prior (denoted by V_{cs} and V_{ca}, respectively). Table 5.4 presents the Chamfer Distance measured on ModelNet. As expected, image-only model (V1) performs worst among these variants. Additionally, we observed that the model will not converge if we adopt V_{cs} without *intra-support* training. (CD is more than 200 and thus are omitted).

5.3.2 Discussion

In this chapter, we introduce a novel triple-branch framework to solve the few-shot single-view point cloud generation problem. We address the few-shot problem by building class-specific shape priors from the support set, modeling class-agnostic shape priors with learnable shape primitives, and decoding the latent vector with a point transformation decoder. In addition, we introduce an episodic training strategy equipped with intra-support augmentation, which avoids finetuning and eliminates the schematic difference between training and testing. Compared to previous fully-supervised methods, we empirically identify that the proposed framework is able to generate a realistic point cloud in the few-shot configuration. Moreover, the proposed framework could also achieve reasonable reconstruction performance in the base classes. Finally, we conclude that combing a metric-space-based network with category-specific shape priors could be an interesting research direction in the future.

CHAPTER 6

ATTENTIONAL FOLDING-BASED POINT CLOUD GENERATION WITH LOCAL SEMANTIC CONSISTENCY

6.1 Approach

In the previous chapters, we focus on the 3D prior knowledge learning and the fusion of features from different domains. In addition to these aspects, the design of the decoder architecture also plays a crucial role for a realistic point cloud generation. Pioneers (Fan et al., 2017; Achlioptas et al., 2018) in this direction adopt vanilla fully connected layers to generate coarse point clouds. However, they inevitably suffer from the scalable issue due to the magnitude of network parameters and convergence speed. To reduce the number of network parameters and generate smoother and denser point clouds, several methods (Yang et al., 2018; Groueix et al., 2018; Deprelle et al., 2019; Lin et al., 2021) embrace the idea of manifold deformation and deform single or multiple canonical 2D grids onto the target surface. We call these models as "folding-based" methods. Generally speaking, these methods approximate the deform function through a shared Multi-Layer Perceptrons (MLP), which dramatically reduces the number of network parameters. However, as shown in Figure 6.1, such design also brings two major limitations: 1) The latent features are combined with the source surface in a brute-force manner, e.g., concatenation, which increases the difficulty of training and thus limits the network's modeling capability; 2) Such network design unconsciously ignores the spatial interactions between each point during the generation process, leading to undesired over-complicated components.

In this chapter, we propose a novel attentional point cloud generation framework to circumvent aforemetioned issues. We have made improvements from two aspects: point cloud decoder and loss function. First, we design our point cloud decoder by following the same principle of TRANSFORMER (Vaswani et al., 2017). Compared with "shared-MLP"

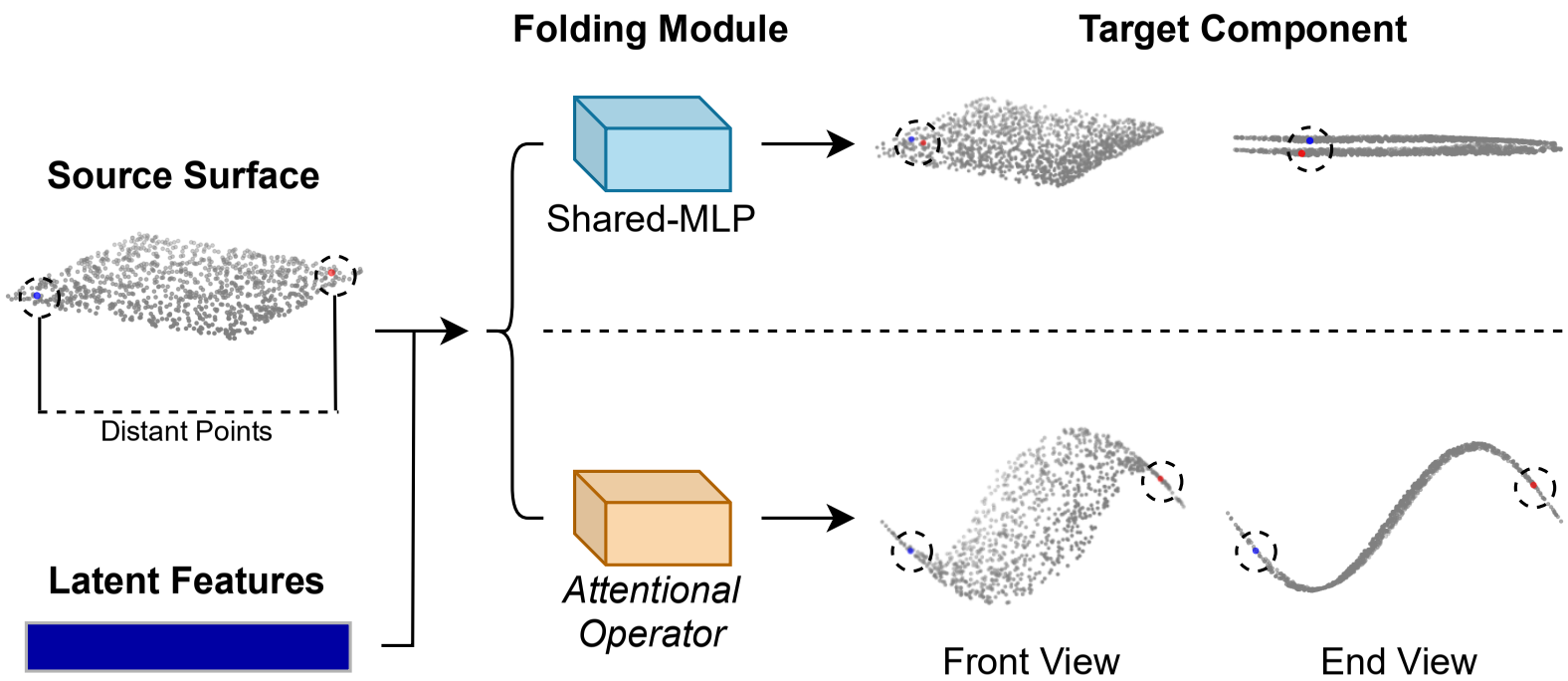

Figure 6.1. The source surface is combined with incoming latent features and transformed onto the target component through a folding module. Ideally, different regions of the source 2-manifold should be deformed onto different areas of the target component. Conventional folding module (shared-MLP) might conduct the deformation in an undesired way because of the negligence of surfaces' global structure. On the other hand, the proposed operator tackles this issu by attentionally considering such information during the generation.

methods that only consider limited local structure around each point, our decoder applies the self-attention mechanism to simultaneously consider the input latent features and the source surface's global context.

Furthermore, we observe that existing point-wise loss function, e.g., Chamfer Distance (CD) or Earth Mover Distance (EMD), cannot faithfully reflect the quality of the generated point clouds (Achlioptas et al., 2018; Lin et al., 2021). We propose a semantic consistency regularizer to address such problem. Specifically, we use a pretrained point cloud feature extractor (e.g., PointNet++ (Qi et al., 2017) or DGCNN (Wang et al., 2019)) to compute the latent features of each generated point and ground truth point. We then conduct an optional sampling process on the generated point cloud to get a small set of target points. For each sampled target point, we compute its corresponding ground truth point via CD or EMD, and calculate the semantic distance between them. Compared to previous researches, our approach considers the global structure of source 2-manifolds during the generation process and avoids the semantic inconsistency induced by Euclidean distance-based loss function.

Let's start with some preliminaries about the folding-based methods and the workflow. We will then dive into two proposed components: 1) Attentional Folding Module, and 2) Local Semantic Consistency in detail.

6.1.1 Preliminary

Manifold deformation provides the theoretical guarantee for folding-based methods regarding point cloud generation. Following the convention of pioneer (Yang et al., 2018), a point cloud can be considered as a sampled discrete subset from the target object's surface. Folding-based methods thus convert the problem to a surface generation problem, which is solved by approximating a diffeomophism Ψ^* between source surface and target surface with a parametric transformation Ψ. Notably, state-of-the-art methods (Groueix et al., 2018; Deprelle et al., 2019; Lin et al., 2021) further boost the performance by decomposing the 3D object

into a set of small surfaces, each having a separate transformation function. With such configuration, we formulate the output point cloud Y_{gen} as the union of all transformed point sets.

$$Y_{gen} = \bigcup_{k=1}^{K} \bigcup_{e \in P_k} \Psi_k(e) \tag{6.1}$$

where K is the number of patches, P_k and Ψ_k are the k-th patch and transformation, respectively. $e \in P_k$ denotes a point sampled from patch P_k.

6.1.2 Framework Overview

In this paper, we aim to generate a complete point cloud from a given source through an attentional folding-based operation. Let $D = \{X^{(i)}, Y_{gt}^{(i)}\}_{i=1}^{M}$ be the paired dataset, where M denotes the cardinality of this dataset, $X^{(i)}$ and $Y_{gt}^{(i)}$ indicate i-th source object and point cloud, respectively. Although a point cloud can conceptually contain infinite points, we set $\|Y_{gt}\| = 2048$ points in this paper to balance between the expressiveness and computational cost (Lin et al., 2021).

As shown in Figure 6.2, we first compute the latent features of a task-specific object X. For instance, if X denotes a 2D image, the task becomes single-view generation; if X represents a point cloud, the task becomes point cloud self-reconstruction; if X merely denotes a partial point cloud, the task becomes point cloud completion. Assume the target point cloud has kn points and the network equips k patches. For each patch, we feed n sampled grids and the latent features to an AFM and collect all the output to form the final point cloud.

The primary component of the loss function is a point-wise Euclidean distance: CD or EMD (Achlioptas et al., 2018). Noted that we formally defined these two losses before and revisit them here for simplicity.

$$\mathcal{L}_{CD} = \sum_{x \in Y_{gen}} \min_{x' \in Y_{gt}} \|x - x'\|_2^2 + \sum_{x' \in Y_{gt}} \min_{x \in Y_{gen}} \|x' - x\|_2^2 \tag{6.2}$$

$$\mathcal{L}_{EMD} = \min_{\phi: Y_{gen} \to Y_{gt}} \sum_{x \in Y_{gt}} \|x - \phi(x)\|_2 \tag{6.3}$$

where $\phi : Y_{gen} \to Y_{gt}$ is a bijection. However, Euclidean distance-based loss alone cannot guarantee a realistic generation (Wu et al., 2018). Therefore we propose to add a novel **L**ocal **S**emantic **C**onsistency (LSC) regularizer. Specifically, we sample m points from Y_{gen} and find their closet point in Y_{gt} via the aforementioned Euclidean metrics. The latent representations of these points are computed via a pre-trained encoder E_{pc} and the semantic consistency is measured by L2 distance.

$$\mathcal{L}_{Semantic} = \frac{1}{|Y'_{gen}|} \sum_{x \in Y'_{gen}} \|E_{pc}(x) - E_{pc}(x')\|_2^2 \tag{6.4}$$

where $Y'_{gen} \subseteq Y_{gen}$ is a sampled subset of the generated point cloud Y_{gen}. Let λ be a weighting factor, the total loss is formulated as:

$$\mathcal{L}_{total} = \mathcal{L}_{CD/EMD} + \lambda \cdot \mathcal{L}_{Semantic} \tag{6.5}$$

6.1.3 Attentional Folding Module

We devise the **A**ttentional-**F**olding-**M**odule (AFM) to consider the global context information of each patch during the surface transformation process.

As shown in Figure 6.3, we can observe that the FC-based methods have no surface/grid structure and purely depend on the latent features. In contrast, the conventional folding-based methods adopt a shared-MLP component and merely considers limited local signal and ignores the global structure of source surface. In contrast, our AFM is able to jointly consider the latent features as well as surface's global context during decoding.

Let $D_{in} = E_{src}(X)$ be the latent features and all source surfaces $\{P\}_{i=1}^{K}$ are 2D unit squares. To generate a point cloud Y_{gen} with $N \cdot K$ points, we first uniformly sample N grids from each patch P_i. Following the same principle in (Vaswani et al., 2017), we map D_{in} and

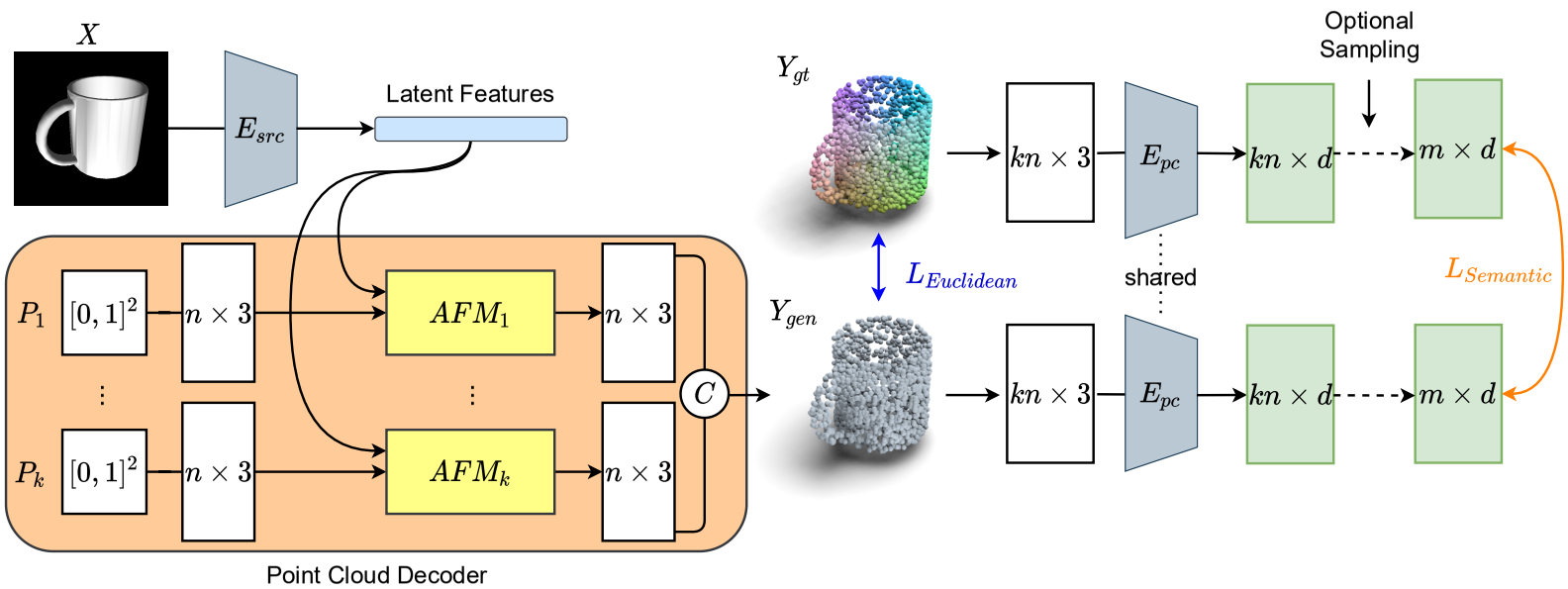

Figure 6.2. Overview. An source encoder E_{src} first extracts the latent features of input X. Noted that X is a 2D image in this plot for demonstration purpose. These features are feed into a point cloud decoder, which contains multiple proposed Attentional Folding Module (AFM). We then collect all points generated by each module to form the final output Y_{gen}. In addition to the conventional Euclidean space point-wise loss (CD/EMD), we further enforce a semantic consistency regularization, $L_{semantic}$, upon m sampled points.

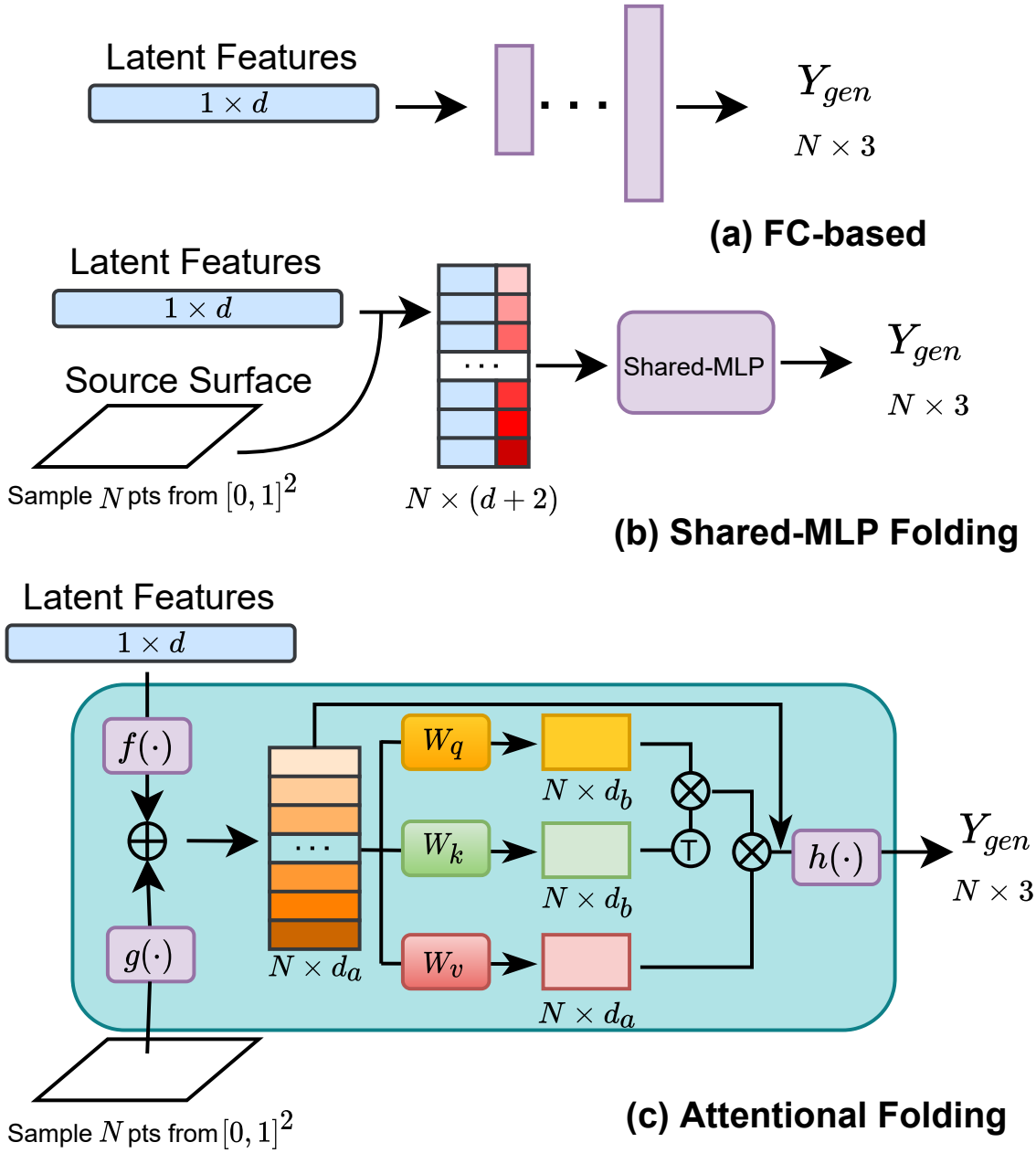

Figure 6.3. Comparison of different point cloud decoders' architecture. For simplicity, folding-based methods (b and c) have 1 patch and the purple blocks in c denote different shared-MLPs.

the sampled grids into the same hidden space and sum them up as the fused features. Let

$f_i(\cdot)$ and $g_i(\cdot)$ be the i-th mapping functions of D_{in} and sampled grids, respectively. The combined features are computed by:

$$D'_i = \bigcup_{e \in P_i} f_i(D_{in}) + g_i(e) \tag{6.6}$$

where $f(\cdot)$ and $g(\cdot)$ are the mapping functions, implemented by simple MLPs, for D_{in} and sampled grids, respectively.

In order to leverage the global surface context and still preserve the permutation invariant property, we adopt the self-attention mechanism to compute an attention weight of each point during the generation. Specifically, we generate Q, K, V matrices by feeding $D'_i \in \mathbb{R}^{N \times d_a}$ into different shared-Linear transformation.

$$\begin{aligned} &Q_i, K_i, V_i = D'_i \cdot (W^i_q, W^i_k, W^i_v) \\ &Q_i, K_i \in \mathbb{R}^{N \times d_b}, V_i \in \mathbb{R}^{N \times d_a} \\ &W^i_q, W^i_k \in \mathbb{R}^{d_a \times d_b}, W^i_v \in \mathbb{R}^{d_a \times d_a} \end{aligned} \tag{6.7}$$

where $W^i_{q,k,v}$ are the parameter matrices for i-th patch, d_b and d_a are the dimension of the KQ and V vectors, respectively. We then compute the attention score via:

$$\text{Attn}(Q_i, K_i, V_i) = \text{softmax}(\frac{Q_i K_i^T}{\sqrt{d_b}})V_i \tag{6.8}$$

These attention scores are further combined with the input D'_i and the result matrix is feed into another simple MLP $h(_i\cdot)$ to generate the final point sets. Therefore, Eq 6.1 can be expanded as:

$$Y_{gen} = \bigcup_{i=1}^{K} h_i(\text{Attn}(Q_i, K_i, V_i) + D') \tag{6.9}$$

In addition, we can conceptually stack multiple AFMs by taking the previous block's output as new source surface and repeatedly injecting D_{in}. Nevertheless, the improvement brought by stacking multiple AFMs is empirically marginal.

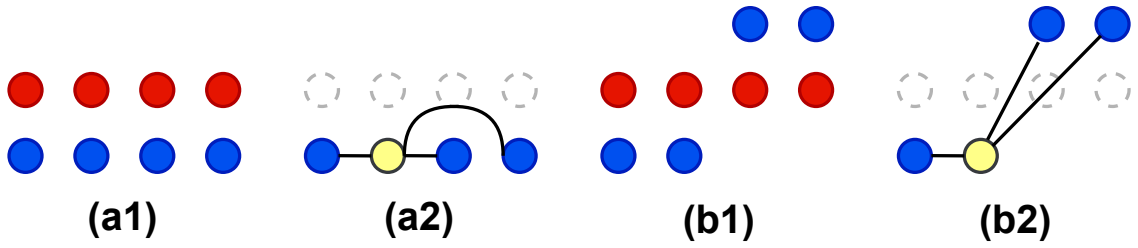

Figure 6.4. **(a1, b1)** Red points are ground truth and blue points are generated. **(a2, b2)** The local structure around the yellow point.

6.1.4 Local Semantic Consistency

The point-wise Euclidean distance cannot faithfully reflect the quality of the generated point cloud (Achlioptas et al., 2018). For example, **a1** and **b1** in Figure 6.4 share the same CD/EMD, yet they represent very different shapes. Therefore, optimizing the network with only point-wise Euclidean distance may lead to sub-optimal solution that achieves good numerical results and unsatisfactory visual quality.

We propose a LSC regularizer to tackle this problem by considering the local structure around selected points (e.g., the yellow point in **a2** and **b2**). Noted that the pretrained point cloud encoder must be a convolutional one (e.g., DGCNN (Wang et al., 2019)), otherwise the ambient signal around each point is ignored. Furthermore, we can compute the LSC upon a subset of the point cloud by applying an optional sampling process (e.g., random sampling and/or farthest point sampling (Fan et al., 2017)). Empirically, using more than 20% points provides similar results and computing LSC globally may even hurt the model's performance.

6.2 Evaluation

We quantitatively and qualitatively evaluate our approach on two challenging tasks: 1) point cloud self-reconstruction and 2) single-view point cloud generation.

6.2.1 Datasets and Baselines

For the self-reconstruction task, we evaluate our method on PointDA (Qin et al., 2019) dataset, which consists of 10 common categories among three famous datasets: ModelNet (Wu et al., 2015), ShapeNet (Chang et al., 2015) and ScanNet (Dai et al., 2017). On the other hand, we adopt 3 categories from the original ModelNet to conduct single-view reconstruction. Follow the same protocol of TDPNet (Lin et al., 2021), we create 12 rendered views by uniformly placing 12 virtual cameras around each CAD model and randomly choose one view as the input RGB image.

All the experiments are trained for 200 epochs with a batch size of 32, using an ADAM optimizer with an initial learning rate of 10^{-3} and $\beta = \{0.9, 0.999\}$. For a fair comparison, we choose both Chamfer Distance and Earth Mover Distance to quantitatively evaluate the performance of all methods.

6.2.2 Point Cloud Self-Reconstruction

We first evaluate our method through a self-reconstruction task. Following SOTA approaches are selected as the baselines: 1) PointFCAE, a simple autoencoder with a MLP decoder; 2) FoldingNet (Yang et al., 2018), which generates point clouds via two consecutive folding operations; 3) AtlasNet (Groueix et al., 2018), which deforms multiple 2D patches onto the target's surface (We adopt 32 patches in the experiment). Noted that all the methods adopt a PointNet (Qi et al., 2017) as their point cloud encoder in this experiment. To better compare with FoldingNet and AtlasNet, we trained our model in two configurations : 1 patch (AttnFold P1) and 32 patches (AttnFold P32).

Table 6.1, Table 6.2, and Table 6.3 present the full results on ModelNet, ShapeNet, and ScanNet self reconstruction, respectively. Although our method does not win every classes, we observed that our model generally outperforms all baselines in term of average performance. In particular, our method is more advantageous regarding EMD without even explicitly

Table 6.1. Quantitative comparison between our method and existing SOTA approaches on ModelNet self reconstruction task. The results of each framework are in the format of CD (x10^3) / EMD (x10^2) for better visualization. Column AVG represents the average performance among all categories. All numbers are obtained from point clouds with 2048 points. AttnFold P1 and AttnFold P32 denote the proposed method trained with 1 patch and 32 patches, respectively. The best results are highlighted in bold and the second bests are highlighted by an underline.

	PointFCAE	FoldingNet	AtlasNet P32	AttnFold P1	AttnFold P32
Bathtub	9.94 / 20.22	10.06 / 40.31	3.54 / 5.35	6.92 / 13.61	**3.38** / **3.60**
Bed	7.13 / 17.98	6.61 / 36.80	3.01 / 3.78	4.85 / 10.32	**2.88** / **3.08**
Bookshelf	7.77 / 16.71	6.75 / 36.23	**3.92** / 4.60	5.49 / 15.06	3.94 / **3.96**
Cabinet	13.38 / 18.09	11.15 / 42.61	3.95 / 5.01	8.22 / 18.81	**3.75** / **3.74**
Chair	10.38 / 16.65	11.80 / 32.16	**3.04** / 5.04	8.22 / 15.80	3.12 / **3.14**
Lamp	29.73 / 28.24	26.89 / 41.02	**6.30** / **12.68**	17.16 / 21.62	6.74 / 13.93
Monitor	7.88 / 20.29	9.27 / 22.04	3.09 / 4.79	5.71 / 9.49	**2.99** / **3.70**
Plant	10.89 / 16.25	14.08 / 26.69	5.81 / 5.98	10.37 / 14.09	**5.72** / **4.42**
Sofa	8.07 / 22.37	8.22 / 36.92	3.81 / 4.23	5.38 / 13.41	**3.22** / **3.70**
Table	11.85 / 20.90	14.06 / 38.25	**2.34** / **6.12**	7.32 / 15.60	2.59 / 6.38
AVG	10.18 / 18.82	10.62 / 33.87	3.68 / 5.31	7.17 / 14.14	**3.59** / **4.81**

Table 6.2. Quantitative comparison between our method and existing SOTA approaches on ShapeNet self reconstruction task. The results of each framework are in the format of CD (x10^3) / EMD (x10^2) for better visualization. Column AVG represents the average performance among all categories. All numbers are obtained from point clouds with 2048 points. AttnFold P1 and AttnFold P32 denote the proposed method trained with 1 patch and 32 patches, respectively. The best results are highlighted in bold and the second bests are highlighted by an underline.

	PointFCAE	FoldingNet	AtlasNet P32	AttnFold P1	AttnFold P32
Bathtub	6.54 / 16.46	4.27 / 9.43	3.84 / 7.98	4.06 / 8.29	**2.89** / **7.87**
Bed	8.17 / 15.95	5.29 / 9.46	**3.06** / **6.24**	5.48 / 9.50	3.60 / 7.91
Bookshelf	4.46 / 14.90	3.56 / 7.37	**2.70** / **5.86**	3.71 / 8.69	2.99 / 6.70
Cabinet	4.18 / 15.05	3.13 / 6.44	**2.47** / **5.91**	3.25 / 6.11	2.65 / 6.28
Chair	5.33 / 12.18	4.28 / 7.99	2.70 / 5.23	4.07 / 8.29	**2.36** / **5.77**
Lamp	6.08 / 11.88	3.80 / 8.76	**2.69** / **5.54**	3.80 / 8.69	2.16 / 7.24
Monitor	4.11 / 13.91	2.98 / 7.67	3.09 / **5.65**	2.97 / 6.72	**2.02** / 6.62
Plant	5.62 / 9.28	4.95 / 6.19	**2.79** / **3.96**	4.81 / 5.91	3.31 / 4.42
Sofa	5.14 / 15.38	3.62 / 8.03	**2.09** / 14.01	3.50 / 7.51	2.65 / **6.36**
Table	6.39 / 14.29	3.78 / 9.17	**1.91** / **5.87**	3.72 / 8.70	2.04 / 6.64
AVG	6.05 / 13.77	3.86 / 8.40	3.37 / 7.06	3.77 / 8.13	**2.33** / **6.45**

Table 6.3. Quantitative comparison between our method and existing SOTA approaches on ScanNet self reconstruction task. The results of each framework are in the format of CD (x10^3) / EMD (x10^2) for better visualization. Column AVG represents the average performance among all categories. All numbers are obtained from point clouds with 2048 points. AttnFold P1 and AttnFold P32 denote the proposed method trained with 1 patch and 32 patches, respectively. The best results are highlighted in bold and the second bests are highlighted by an underline.

	PointFCAE	FoldingNet	AtlasNet P32	AttnFold P1	AttnFold P32
Bathtub	13.54 / 31.74	30.20 / 29.79	**3.21** / 17.71	8.23 / 28.71	3.40 / **11.72**
Bed	12.70 / 29.86	25.88 / 29.90	**3.03** / 15.29	6.43 / 24.30	3.11 / **10.15**
Bookshelf	12.42 / 27.26	27.71 / 23.84	**4.05** / 14.88	7.01 / 22.04	4.11 / **10.33**
Cabinet	15.34 / 28.37	34.52 / 23.10	**3.17** / 15.59	6.32 / 24.31	3.32 / **10.91**
Chair	15.93 / 29.93	33.89 / 27.81	**3.75** / 15.93	7.97 / 25.78	**3.75** / **10.34**
Lamp	17.83 / 32.21	38.48 / 27.71	**4.31** / 17.39	9.69 / 27.72	4.68 / **12.27**
Monitor	12.86 / 28.26	32.89 / 25.87	**3.13** / 14.90	5.81 / 21.63	3.42 / **10.01**
Plant	13.62 / 22.89	29.55 / 22.54	**4.86** / 11.82	8.45 / 19.60	4.97 / **7.73**
Sofa	13.46 / 22.80	26.69 / 28.89	**3.63** / 15.97	7.20 / 24.77	3.69 / **10.53**
Table	12.71 / 31.43	30.28 / 30.44	3.27 / 16.99	5.97 / 26.04	**3.05** / **12.19**
AVG	14.68 / 29.71	31.84 / 27.45	**3.60** / 15.93	7.24 / 25.01	3.69 / **10.72**

optimizing it. For example, we are able to reduce EMD from the second best 5.31 to 4.81 on ModelNet dataset. Moreover, look at the performance of FoldingNet and AttnFold P1, we further conclude that employing an attentional folding operator not only reduces model size, but also achieves better performance.

The qualitative results in Figure 6.5 further demonstrates the perceptual advantage of our method. In comparison, our method is able to generate more realistic and smoother point cloud whereas other methods are prone to generate sparse and blurry results. Specifically, PointFCAE and FoldingNet fail to generate a complete shape and AtlasNet tends to provide round and blurry borderlines (e.g., bed head)

6.2.3 Single-View Point Cloud Reconstruction

As an advanced folding module, our framework is capable of solving a more challenging sing-view point cloud reconstruction problem, whose input is a single 2D image. For the

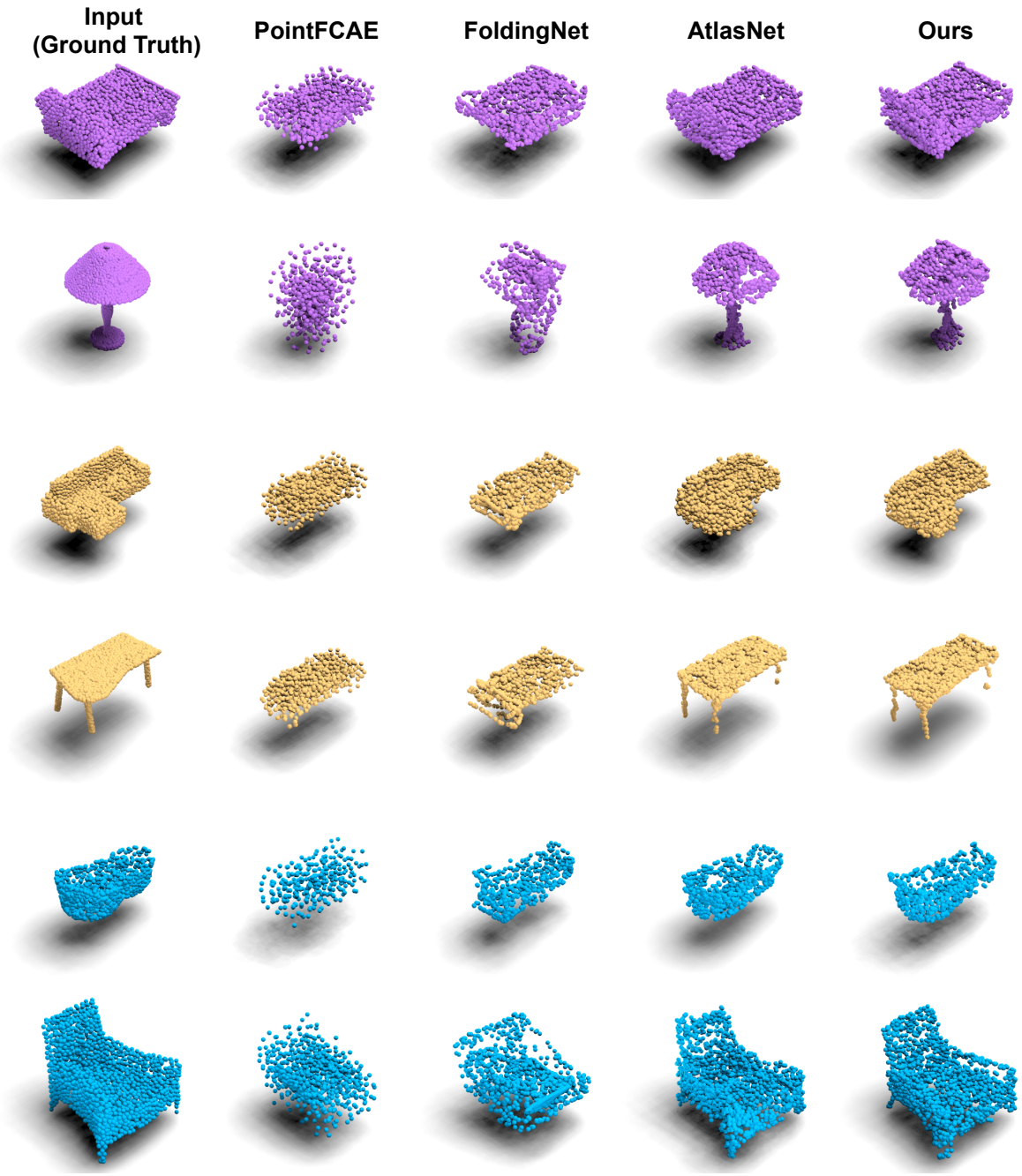

Figure 6.5. Qualitative comparison on ModelNet self-reconstruction. From top to bottom: Bed, Lamp, Sofa, Table, Bathtub, and Chair.

FC-absed baselines, we replace PointFCAE with a more sophisticate framework, PSGN (Fan et al., 2017), that achieves better reconstruction by adopting a de-convolutional branch. To

Table 6.4. Quantitative comparison of single-view reconstruction on 3 categories of *ModelNet*. Results are reported in the format of CD (x10^3) / EMD (x10^2). Best results are bolded.

	Airplane	Bed	Chair
PSGN	6.12 / 20.89	10.42 / 27.05	14.07 / 24.98
FoldingNet	6.03 / 11.36	8.58 / 7.74	9.30 / 23.76
AtlasNet	5.46 / 9.69	7.54 / 6.97	8.89 / 8.67
AttnFold P32	**4.93** / **8.28**	**6.29** / **4.63**	**7.25** / **6.88**

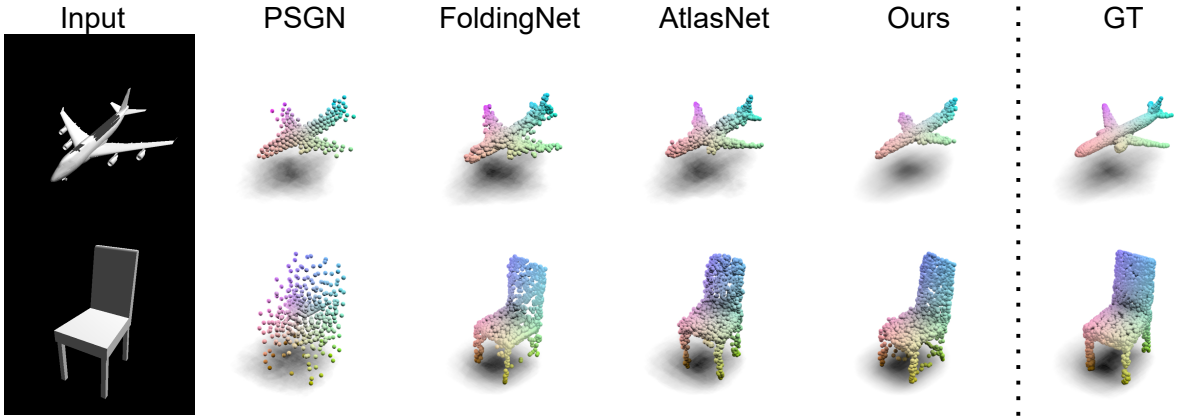

Figure 6.6. Qualitative comparison on ModelNet single-view reconstruction. Top: airplane. Bottom: chair.

avoid the discrepancy caused by image encoder, a pretrained ResNet-18 (He et al., 2016) is adopted as the feature extractor.

Table 6.4 shows the performance of competing methods on three categories of ModelNet: airplane, bed, and chair. We found that our method outperforms all baselines regarding CD and EMD in a large margin. Moreover, Figure 6.6 presents a visual comparison among all competitors. It's interesting to see that our method tends to focus on the main component of the target shape, provides a smoother surface (e.g., airplane body and chair back), and ignores distant fine-grain details (e.g., airplane tail). Considering the difficulty of single-view reconstruction, we conclude that our method achieves better performance, although it's still far from real-world applications.

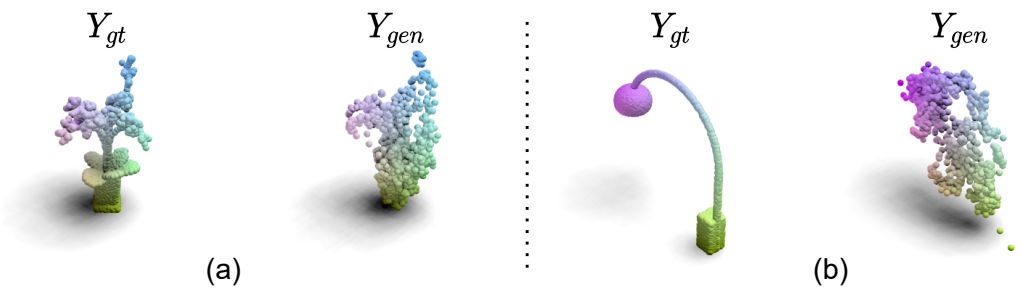

Figure 6.7. Samples of failure cases on ModelNet self-reconstruction task. Y_{gt} and Y_{gen} denote ground truth and the generated point cloud, respectively. Left: plant. Right: lamp.

Table 6.5. ModelNet self-reconstruction performance upon different number of patches. Results share the same format in Table 6.4

	Bathtub	Chair	Monitor	Table	**Avg**
P=1	6.92 / 13.16	8.22 / 15.80	5.71 / 9.49	7.32 / 15.60	7.17 / 14.14
P=4	4.83 / 10.58	4.87 / 8.01	4.07 / 6.91	4.13 / 8.71	4.97 / 6.92
P=8	4.20 / 7.22	4.03 / 7.94	3.68 / 5.75	3.05 / 8.40	4.31 / 6.63
P=16	3.62 / 4.55	3.36 / 5.11	3.13 / 4.74	2.75 / 7.23	3.76 / 5.36
P=32	3.38 / 3.60	3.12 / 3.14	2.99 / 3.70	2.59 / 6.38	3.59 / 4.81

6.3 Ablation Studies and Discussion

In this section, we will first verify the effectiveness of the proposed method under various network architectures. Later, ablation experiments are conducted to investigate the impact of LSC regularizer hyper-parameters. Finally, we will present some failure cases and discuss the method's limitation.

6.3.1 Network Architectures

Table 6.5 reports the ModelNet self-reconstruction performance regarding different number of patches. As exploited in previous literatures (Groueix et al., 2018; Lin et al., 2021), we observed the same pattern that increasing the number of patches gradually improves the model performance and the benefits diminished on large patch numbers. Moreover, such observation shows that the proposed method is applicable to various network architectures.

Table 6.6. Average self-reconstruction performance on ModelNet with different values of LSC weight factor λ (Equation 6.5). CD and EMD are multiplied by 10^2 and 10^3, respectively.

$\lambda =$		0.0	0.1	0.5	1.0	2.0	5.0
AVG	CD	3.73	3.81	3.68	3.59	3.88	4.60
	EMD	5.17	5.99	5.13	4.81	4.81	5.03

Table 6.7. Average self-reconstruction performance on ModelNet with different number of sampling points ρ. Same format as Table 6.6

$\rho =$		0	200	400	800	1600	2048
AVG	CD	3.74	3.59	3.65	3.65	3.71	3.85
	EMD	5.19	4.81	4.80	4.84	5.33	5.42

6.3.2 Impact of Hyper-parameters

We then study how different hyper-parameters affect the training outcome. Specifically, we played with following hyper-parameters: 1) λ, the weight factor of LSC regularizer and 2) ρ, the number of points to be sampled in the LSC calculation. In order to conduct a fair comparison, we set $\lambda = 1.0$ when tuning ρ and set $\rho = 200$ when tuning λ.

Table 6.6 and Table 6.7 present the model performance on different λ and ρ, respectively. Interestingly, we observed that increasing the weight of LSC does not always lead to better results. For example, both CD and EMD start to fall after $\lambda = 1.0$. On the other hand, the number of sampling point ρ tend to provide a more robust improvement when $\rho \in [200, 800]$. Nevertheless, adopting a very large ρ will also slightly downgrade the performance.

6.3.3 Failure Cases and Limitations

Figure 6.7 shows two typical failure cases of our method. As we can see, the proposed method is unable to generate some shapes with asymmetric (plant) and thin structures (lamp) correctly. We believe this phenomenon is potentially caused by the continuity of source surfaces, which might be alleviated by using a more flexible and discrete topological object.

Moreover, another limitation of the proposed method is its sensitivity to the selection of hyper-parameters. As mentioned in the previous section, hyper-parameters (*e.g.,* λ and ρ) have to be carefully tuned to achieve promising results. These limitations might affect the applicability of the proposed method in real-world scenarios.

6.3.4 Discussion

In this chapter, we propose an advanced folding-based decoder that attentionally generates point clouds from arbitrary latent features. Specifically, it generates point clouds by a popular self-attention mechanism that weights the point from each patch in an interim space. In addition, a semantic consistency regularizer is introduced to further improve the framework's modeling capability. Extensive experiments on different tasks and datasets show that our method achieves better results than previous state-of-the-art folding-based approaches. Moreover, we empirically investigate the impact of various hyper-parameters and network architectures, which further demonstrates the effectiveness of the proposed method.

CHAPTER 7

CONCLUSION AND FUTURE WORK [1] [2]

In this chapter, we will draw a conclusion and discuss our future work. We will first start with our point cloud generation frameworks, TDPNet and L-SHAP, that adopt explicit 3D shape prior, and later focus on the FSPG and AttnFold that generates point cloud in the few-shot setting and conducts advanced folding operation, respectively.

7.1 Point Cloud Generation with Explicit 3D Priors

In TDPNet, we introduce a unified framework for generating point clouds from arbitrary information source (e.g., single view image). Our method achieves superior performance compared to several state-of-the-art baselines both quantitatively and qualitatively. Specifically, TDPNet is a 2-branch point cloud generation framework that simultaneously considers 3D shape priors and other information sources. The network is trained in two stages. In the first stage, we warm up the network by solving a point cloud self-reconstruction problem. Noted that only point clouds are available in this stage, thus an all-zero dummy vector is adopted as the placeholder of other information sources. In the second stage, we utilize the pretrained point cloud feature extractor to compute the KMeans centroids of a given class as the 3D prototype features. These 3D prototype features are then combined with incoming features (e.g., latent image representation) and be fed into a hierarchical manifold decoder to democratically construct the final point cloud. Therefore, our approach bridges the gap between 2D and 3D features by introducing a flexible 3D prototype mechanism.

---

[1]This chapter contains material previously published as: Yu Lin, Yigong Wang, Yifan Li, Zhuoyi Wang, Yang Gao, and Latifur Khan. "Single View Point Cloud Generation via Unified 3D Prototype". In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 3, pp. 2064-2072. 2021

[2]This chapter contains material previously published as: Yu Lin, Jinghui Guo, Yang Gao, Yifan Li, Zhuoyi Wang, and Latifur Khan. "Generating Point Cloud from Single Image in The Few Shot Scenario". In *Proceedings of the 29th ACM International Conference on Multimedia*, pp. 2834-2842. 2021

Nevertheless, KMeans is not the only clustering algorithm and its unstable behavior might hurt the model's performance. It would be interesting to investigate the impact of other clustering methods (Breen et al., 2002; Awad et al., 2004; Petrushin and Khan, 2007), such as DBSCAN, OPTICS, and Gaussian Mixture, since they produce quite different clusters compared to KMeans. Moreover, recently proposed deep clustering approaches (Caron et al., 2018) might be a good choice because we can train the network in an end-to-end manner, instead of a two-stage pipeline, by using such a method. In addition to the clustering algorithm, inventing a more sophisticated feature infusion method is another potential direction. Like what we proposed in the AttnFold, we can project other information sources and 3D prototype features into a joint feature space and perform the element-wise summation.

7.2 Point Cloud Generation with Learnable 3D Priors

As we observed that KMeans clustering is sensitive to the initialization seeds and produces unstable centroids, we proposed to learn the 3D shape priors from training data in L-SHAP. In this framework, we introduced a new component, named learnable shape primitive, to encode the 3D shape prior information. Specifically, we consider each shape primitive as an initial point set and a transformation function. The proposed shape primitive can either be a continuous one or a discrete one, based on the assumption of the initial point set. Continuous shape primitive is a good choice if dense point cloud reconstruction is needed, since arbitrary points can be sampled from this data structure. On the other hand, discrete shape primitive is amenable if the applications demand a more realistic yet size-fixed point cloud. Experiments on both shape primitives demonstrate that our model outperforms other state-of-the-art methods.

Despite its success, we observed that the learned shape primitives are not interpretable compared with clustering centroid prototypes in TDPNet. One possible cause is that the

network is trained in an end-to-end manner, and the proposed shape primitives are the intermediate results that have to incorporate with other information resources to generate the final results (Yen et al., 2002; Goodman et al., 2010). As interpretability of deep learning becomes more and more important, learning interpretable 3D shape priors would be a potential direction. For example, instead of directly predicting the final point cloud, what if we can predict the affine transformation matrix and apply it to the original point set?

7.3 Few-shot Point Cloud Generation

Inspired by TDPNet and L-SHAP, we propose a triple-branch point cloud generation framework named FSPG. Following the same protocol of ProtoNet, we compute the element-wise average of support point clouds' latent representation as class-specific shape prior. We then adopt the same network architecture proposed in L-SHAP to encode the class-agnostic shape prior information across different classes. Class-specific and class-agnostic shape priors are then incorporated with a given latent image representation to generate the final point cloud. Moreover, due to the lack of training pairs, we further introduce a intra-support augmentation method to incorporate with 1-way-k-shot episodic training. Compared to previous few-shot generation methods, we empirically demonstrate the effectiveness of the proposed method in various datasets.

During the experiment, we observed that the model cannot provide satisfying results if the point clouds in the support set are very different from the target point cloud (Masud et al., 2007). This phenomenon is quite common under the few-shot setting, and failing to generate realistic point clouds might impair the usefulness of the proposed method. The future work of FSPG will focus on solving this issue.

7.4 Advanced Folding Operation

In the AttnFold, we introduce an advanced folding-based decoder that adopts Transformer architecture to attentionally generate point clouds. Specifically, we project image features and coordinates of sampled points from 2D planes (as 3D priors) to the same latent space and compute their sum as our latent features. These latent features are then passed to three shared-Linear functions to compute the attention score of each pair of two points. Such attention scores allow each point to be affected by every other point in the same 2-manifold, which avoids the undesired over-complicated shape component. In addition to the attentional folding decoder, we further propose a local semantic consistency regularizer that considers the semantic discrepancy between the generated point cloud and ground truth to alleviate the visual quality issues caused by the point-wise loss functions.

We observed that the proposed method tend to create smooth and dense major shape component and leaves small parts blurry. Moreover, the proposed method also fails to generate some asymmetric and thin structures correctly. Therefore, the future work of AttnFold would be investigating a more sophisticated network architecture and trying a more semantic regularizer rather than sticking with point-wise loss functions, such as Chamfer Distance or Earth Mover Distance. We will work on this project in the future.

REFERENCES

Abedin, M., S. Nessa, L. Khan, and B. Thuraisingham (2006). Detection and resolution of anomalies in firewall policy rules. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pp. 15–29. Springer.

Abrol, S. and L. Khan (2010a). Tweethood: Agglomerative clustering on fuzzy k-closest friends with variable depth for location mining. In *2010 IEEE Second International Conference on Social Computing*, pp. 153–160. IEEE.

Abrol, S. and L. Khan (2010b). Twinner: understanding news queries with geo-content using twitter. In *Proceedings of the 6th Workshop on Geographic information Retrieval*, pp. 1–8.

Achlioptas, P., O. Diamanti, I. Mitliagkas, and L. Guibas (2018). Learning representations and generative models for 3d point clouds. In *International conference on machine learning*, pp. 40–49. PMLR.

Ahmed, E., A. Saint, A. E. R. Shabayek, K. Cherenkova, R. Das, G. Gusev, D. Aouada, and B. Ottersten (2018). A survey on deep learning advances on different 3d data representations. *arXiv preprint arXiv:1808.01462*.

Awad, M., L. Khan, F. Bastani, and I.-L. Yen (2004). An effective support vector machines (svms) performance using hierarchical clustering. In *16th IEEE international conference on tools with artificial intelligence*, pp. 663–667. IEEE.

Awad, M., L. Khan, and B. Thuraisingham (2008). Predicting www surfing using multiple evidence combination. *The VLDB Journal 17*(3), 401–417.

Awad, M. A. and L. R. Khan (2007). Web navigation prediction using multiple evidence combination and domain knowledge. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans 37*(6), 1054–1062.

Blanc, T., M. El Beheiry, C. Caporal, J.-B. Masson, and B. Hajj (2020). Genuage: visualize and analyze multidimensional single-molecule point cloud data in virtual reality. *Nature Methods 17*(11), 1100–1102.

Blinn, J. F. (1977). Models of light reflection for computer synthesized pictures. In *Proceedings of the 4th annual conference on Computer graphics and interactive techniques*, pp. 192–198.

Breen, C., L. Khan, and A. Ponnusamy (2002). Image classification using neural networks and ontologies. In *Proceedings. 13th International Workshop on Database and Expert Systems Applications*, pp. 98–102. IEEE.

Cardenas-Garcia, J., H. Yao, and S. Zheng (1995). 3d reconstruction of objects using stereo imaging. *Optics and Lasers in Engineering 22*(3), 193–213.

Caron, M., P. Bojanowski, A. Joulin, and M. Douze (2018). Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 132–149.

Chang, A. X., T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. (2015). Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*.

Cheung, K., S. Baker, and T. Kanade (2003). Shape-from-silhouette of articulated objects and its use for human body kinematics estimation and motion capture. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, Volume 1, pp. I–I. IEEE.

Chibane, J., T. Alldieck, and G. Pons-Moll (2020). Implicit functions in feature space for 3d shape reconstruction and completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6970–6981.

Choy, C. B., D. Xu, J. Gwak, K. Chen, and S. Savarese (2016). 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision*, pp. 628–644. Springer.

Csáji, B. C. et al. (2001). Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary 24*(48), 7.

Cui, Y., R. Chen, W. Chu, L. Chen, D. Tian, Y. Li, and D. Cao (2021). Deep learning for image and point cloud fusion in autonomous driving: A review. *IEEE Transactions on Intelligent Transportation Systems*.

Dai, A., A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner (2017). Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5828–5839.

de Souza Cardoso, L. F., F. C. M. Q. Mariano, and E. R. Zorzal (2020). A survey of industrial augmented reality. *Computers & Industrial Engineering 139*, 106159.

Debnath, B., M. Solaimani, M. A. G. Gulzar, N. Arora, C. Lumezanu, J. Xu, B. Zong, H. Zhang, G. Jiang, and L. Khan (2018). Loglens: A real-time log analysis system. In *2018 IEEE 38th international conference on distributed computing systems (ICDCS)*, pp. 1052–1062. IEEE.

Deprelle, T., T. Groueix, M. Fisher, V. Kim, B. Russell, and M. Aubry (2019). Learning elementary structures for 3d shape generation and matching. In *Advances in Neural Information Processing Systems*, pp. 7433–7443.

Esteban, C. H. and F. Schmitt (2002). Multi-stereo 3d object reconstruction. In *Proceedings. First International Symposium on 3D Data Processing Visualization and Transmission*, pp. 159–166. IEEE.

Fan, H., H. Su, and L. J. Guibas (2017). A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 605–613.

Fei-Fei, L., R. Fergus, and P. Perona (2006). One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence 28*(4), 594–611.

Gadelha, M., S. Maji, and R. Wang (2017). 3d shape induction from 2d views of multiple objects. In *2017 International Conference on 3D Vision (3DV)*, pp. 402–411. IEEE.

Gadelha, M., R. Wang, and S. Maji (2018). Multiresolution tree networks for 3d point cloud processing. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 103–118.

Geiger, A., J. Ziegler, and C. Stiller (2011). Stereoscan: Dense 3d reconstruction in real-time. In *2011 IEEE intelligent vehicles symposium (IV)*, pp. 963–968. Ieee.

Genova, K., F. Cole, D. Vlasic, A. Sarna, W. T. Freeman, and T. Funkhouser (2019). Learning shape templates with structured implicit functions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7154–7164.

Goodman, L. A., K. F. Smyth, and V. Banyard (2010). Beyond the 50-minute hour: increasing control, choice, and connections in the lives of low-income women. *American Journal of Orthopsychiatry 80*(1), 3.

Gotsman, C., X. Gu, and A. Sheffer (2003). Fundamentals of spherical parameterization for 3d meshes. In *ACM SIGGRAPH 2003 Papers*, pp. 358–363.

Groueix, T., M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry (2018). A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 216–224.

Guo, J., A. Ersen, Y. Gao, Y. Lin, L. Khan, and M. Yavuz (2020). Prediction of plantar shear stress distribution by conditional gan with attention mechanism. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 770–780. Springer.

Han, X.-F., H. Laga, and M. Bennamoun (2019). Image-based 3d object reconstruction: State-of-the-art and trends in the deep learning era. *IEEE transactions on pattern analysis and machine intelligence 43*(5), 1578–1604.

He, K., X. Zhang, S. Ren, and J. Sun (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Hua, B.-S., M.-K. Tran, and S.-K. Yeung (2018). Pointwise convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 984–993.

Jiang, L., S. Shi, X. Qi, and J. Jia (2018). Gal: Geometric adversarial loss for single-view 3d-object reconstruction. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 802–816.

Jin, Y., L. Khan, L. Wang, and M. Awad (2005). Image annotations by combining multiple evidence & wordnet. In *Proceedings of the 13th annual ACM international conference on Multimedia*, pp. 706–715.

Kaiser, A., J. A. Ybanez Zepeda, and T. Boubekeur (2019). A survey of simple geometric primitives detection methods for captured 3d data. In *Computer Graphics Forum*, Volume 38, pp. 167–196. Wiley Online Library.

Kato, H. and T. Harada (2019). Learning view priors for single-view 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9778–9787.

Khan, L. and D. McLeod (2000). Audio structuring and personalized retrieval using ontologies. In *Proceedings IEEE Advances in Digital Libraries 2000*, pp. 116–126. IEEE.

Kingma, D. P. and M. Welling (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Lai, K., L. Bo, X. Ren, and D. Fox (2011). A large-scale hierarchical multi-view rgb-d object dataset. In *2011 IEEE international conference on robotics and automation*, pp. 1817–1824. IEEE.

Lavee, G., L. Khan, and B. Thuraisingham (2007). A framework for a video analysis tool for suspicious event detection. *Multimedia Tools and Applications 35*(1), 109–123.

LeCun, Y., Y. Bengio, and G. Hinton (2015). Deep learning. *nature 521*(7553), 436–444.

Li, X., S. Du, G. Li, and H. Li (2020). Integrate point-cloud segmentation with 3d lidar scan-matching for mobile robot localization and mapping. *Sensors 20*(1), 237.

Li, Y., X. Wu, Y. Chrysathou, A. Sharf, D. Cohen-Or, and N. J. Mitra (2011). Globfit: Consistently fitting primitives by discovering global relations. In *ACM SIGGRAPH 2011 papers*, pp. 1–12.

Lin, C.-H., C. Kong, and S. Lucey (2018). Learning efficient point cloud generation for dense 3d object reconstruction. In *proceedings of the AAAI Conference on Artificial Intelligence*, Volume 32.

Lin, Y., J. Guo, Y. Gao, Y.-f. Li, Z. Wang, and L. Khan (2021). Generating point cloud from single image in the few shot scenario. In *Proceedings of the 29th ACM International Conference on Multimedia*, pp. 2834–2842.

Lin, Y., Y. Wang, Y.-F. Li, Z. Wang, Y. Gao, and L. Khan (2021). Single view point cloud generation via unified 3d prototype. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Volume 35, pp. 2064–2072.

Liu, X., Z. Han, X. Wen, Y.-S. Liu, and M. Zwicker (2019). L2g auto-encoder: Understanding point clouds by local-to-global reconstruction with hierarchical self-attention. In *Proceedings of the 27th ACM International Conference on Multimedia*, pp. 989–997.

Lobay, A. and D. A. Forsyth (2006). Shape from texture without boundaries. *International Journal of Computer Vision 67*(1), 71–91.

Mandikal, P., K. Navaneet, M. Agarwal, and R. V. Babu (2018). 3d-lmnet: Latent embedding matching for accurate and diverse 3d point cloud reconstruction from a single image. *arXiv preprint arXiv:1807.07796*.

Martins, A. and R. Astudillo (2016). From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International Conference on Machine Learning*, pp. 1614–1623. PMLR.

Masci, J., D. Boscaini, M. Bronstein, and P. Vandergheynst (2015). Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the IEEE international conference on computer vision workshops*, pp. 37–45.

Masud, M. M., J. Gao, L. Khan, J. Han, and B. Thuraisingham (2009). A multi-partition multi-chunk ensemble technique to classify concept-drifting data streams. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 363–375. Springer.

Masud, M. M., J. Gao, L. Khan, J. Han, and B. Thuraisingham (2010). Classification and novel class detection in data streams with active mining. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 311–324. Springer.

Masud, M. M., L. Khan, and B. Thuraisingham (2007). A hybrid model to detect malicious executables. In *2007 IEEE International Conference on Communications*, pp. 1443–1448. IEEE.

Masud, M. M., L. R. Khan, B. M. Thuraisingham, Q. Chen, J. Gao, and J. Han (2015, October 20). Systems and methods for detecting a novel data class. US Patent 9,165,051.

Mejjati, Y. A., C. Richardt, J. Tompkin, D. Cosker, and K. I. Kim (2018). Unsupervised attention-guided image-to-image translation. In *Advances in Neural Information Processing Systems*, pp. 3693–3703.

Michalkiewicz, M., S. Parisot, c. S. Tsogkas, M. Baktashmotlagh, A. Eriksson, and E. Belilovsky (2020). Few-shot single-view 3-d object reconstruction with compositional priors. *arXiv preprint arXiv:2004.06302*.

Monti, F., D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein (2017). Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5115–5124.

Musgrave, K., S. Belongie, and S.-N. Lim (2020). Pytorch metric learning.

Nessa, S., M. Abedin, W. E. Wong, L. Khan, and Y. Qi (2008). Software fault localization using n-gram analysis. In *International Conference on Wireless Algorithms, Systems, and Applications*, pp. 548–559. Springer.

Petrushin, V. A. and L. Khan (2007). *Multimedia data mining and knowledge discovery*, Volume 521. Springer.

Pontes, J. K., C. Kong, S. Sridharan, S. Lucey, A. Eriksson, and C. Fookes (2018). Image2mesh: A learning framework for single image 3d reconstruction. In *Asian Conference on Computer Vision*, pp. 365–381. Springer.

Praun, E. and H. Hoppe (2003). Spherical parametrization and remeshing. *ACM Transactions on Graphics (TOG) 22*(3), 340–349.

Qi, C. R., H. Su, K. Mo, and L. J. Guibas (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660.

Qi, C. R., L. Yi, H. Su, and L. J. Guibas (2017). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pp. 5099–5108.

Qin, C., H. You, L. Wang, C.-C. J. Kuo, and Y. Fu (2019). Pointdan: A multi-scale 3d domain adaption network for point cloud representation. *Advances in Neural Information Processing Systems 32*.

Rezende, D. J., S. A. Eslami, S. Mohamed, P. Battaglia, M. Jaderberg, and N. Heess (2016). Unsupervised learning of 3d structure from images. In *Advances in neural information processing systems*, pp. 4996–5004.

Schmid, K., H. Hirschmüller, A. Dömel, I. Grixa, M. Suppa, and G. Hirzinger (2012). View planning for multi-view stereo 3d reconstruction using an autonomous multicopter. *Journal of Intelligent & Robotic Systems 65*(1), 309–323.

Schnabel, R., P. Degener, and R. Klein (2009). Completion and reconstruction with primitive shapes. In *Computer Graphics Forum*, Volume 28, pp. 503–512. Wiley Online Library.

Schnabel, R., R. Wahl, and R. Klein (2007). Efficient ransac for point-cloud shape detection. In *Computer graphics forum*, Volume 26, pp. 214–226. Wiley Online Library.

Sharma, C. and M. Kaul (2020). Self-supervised few-shot learning on point clouds. *NeurIPS*.

Sheffer, A., E. Praun, K. Rose, et al. (2007). Mesh parameterization methods and their applications. *Foundations and Trends® in Computer Graphics and Vision 2*(2), 105–171.

Simonovsky, M. and N. Komodakis (2017). Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3693–3702.

Snell, J., K. Swersky, and R. Zemel (2017). Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pp. 4077–4087.

Stets, J. D., Y. Sun, W. Corning, and S. W. Greenwald (2017). Visualization and labeling of point clouds in virtual reality. In *SIGGRAPH Asia 2017 Posters*, pp. 1–2.

Su, H., S. Maji, E. Kalogerakis, and E. Learned-Miller (2015). Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pp. 945–953.

Sung, M., H. Su, R. Yu, and L. J. Guibas (2018). Deep functional dictionaries: Learning consistent semantic structures on 3d models from functions. In *Advances in Neural Information Processing Systems*, pp. 485–495.

Tu, M., P. Li, I.-L. Yen, B. M. Thuraisingham, and L. Khan (2008). Secure data objects replication in data grid. *IEEE Transactions on dependable and secure computing 7*(1), 50–64.

Tulsiani, S., H. Su, L. J. Guibas, A. A. Efros, and J. Malik (2017). Learning shape abstractions by assembling volumetric primitives. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2635–2643.

Tulsiani, S., T. Zhou, A. A. Efros, and J. Malik (2017). Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2626–2634.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin (2017). Attention is all you need. *Advances in neural information processing systems 30*.

Vinyals, O., C. Blundell, T. Lillicrap, D. Wierstra, et al. (2016). Matching networks for one shot learning. *Advances in neural information processing systems 29*, 3630–3638.

Wallace, B. and B. Hariharan (2019). Few-shot generalization for single-image 3d reconstruction via priors. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3818–3827.

Wang, C., M. Cheng, F. Sohel, M. Bennamoun, and J. Li (2019). Normalnet: A voxel-based cnn for 3d object classification and retrieval. *Neurocomputing 323*, 139–147.

Wang, L., L. Liu, and L. Khan (2004). Automatic image annotation and retrieval using subspace clustering algorithm. In *Proceedings of the 2nd ACM international workshop on Multimedia databases*, pp. 100–108.

Wang, N., Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang (2018). Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 52–67.

Wang, Y., Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon (2019). Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog) 38*(5), 1–12.

Wang, Y., Q. Yao, J. T. Kwok, and L. M. Ni (2020). Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys (CSUR) 53*(3), 1–34.

Wu, J., C. Zhang, X. Zhang, Z. Zhang, W. T. Freeman, and J. B. Tenenbaum (2018). Learning shape priors for single-view 3d completion and reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 646–662.

Wu, Z., S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao (2015). 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1912–1920.

Yang, G., X. Huang, Z. Hao, M.-Y. Liu, S. Belongie, and B. Hariharan (2019). Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4541–4550.

Yang, Y., C. Feng, Y. Shen, and D. Tian (2018). Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 206–215.

Yen, I.-L., J. Goluguri, F. Bastani, L. Khan, and J. Linn (2002). A component-based approach for embedded software development. In *Proceedings Fifth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing. ISIRC 2002*, pp. 402–410. IEEE.

Yue, X., B. Wu, S. A. Seshia, K. Keutzer, and A. L. Sangiovanni-Vincentelli (2018). A lidar point cloud generator: from a virtual world to autonomous driving. In *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*, pp. 458–464.

Zaheer, M., S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola (2017). Deep sets. *Advances in neural information processing systems 30*.

Zhang, R., P.-S. Tsai, J. E. Cryer, and M. Shah (1999). Shape-from-shading: a survey. *IEEE transactions on pattern analysis and machine intelligence 21*(8), 690–706.

Zhao, Y., T. Birdal, H. Deng, and F. Tombari (2019). 3d point capsule networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1009–1018.

Zou, C., E. Yumer, J. Yang, D. Ceylan, and D. Hoiem (2017). 3d-prnn: Generating shape primitives with recurrent neural networks. In *Proc. of ICCV*.

BIOGRAPHICAL SKETCH

Yu Lin received his Bachelor of Engineering degree in software engineering from the University of Electronic Science and Technology of China (UESTC) in 2016. Later that year, he decided to engage in advanced studies to further his knowledge in machine learning and 3D computer vision. Therefore, he joined The University of Texas at Dallas (UTD) to pursue his PhD degree in computer science. During his PhD studies, Yu was a member of the Big Data Analytics and Management Lab and worked as a research assistant under the supervision of Prof. Latifur Khan. His research concentrates on machine learning, deep leanring, and 2D and 3D data generation.

CURRICULUM VITAE

Yu Lin

March, 2022

Contact Information:

Department of Computer Science
The University of Texas at Dallas
800 W. Campbell Rd.
Richardson, TX 75080-3021, U.S.A.

Email: `yxl163430@utdallas.edu`

Educational History:

BE, Software Engineering, University of Electronic Science and Technology of China, 2016
MS, Computer Science, University of Texas at Dallas, 2022
PhD, Computer Science, University of Texas at Dallas, 2022

Enhancing Point Cloud Generation from Various Information Sources by Applying Geometry-aware Folding Operation
PhD Dissertation
Computer Science Department, The University of Texas At Dallas
Advisors: Dr. Latifur Khan

Employment History:

Software Engineer Intern, Meta, Inc, May 2021 – August 2021
Research Assistant, The University of Texas at Dallas, June 2019 – present
Teaching Assistant, The University of Texas at Dallas, September 2018 – May 2019

Technical Skills:

Programming: Python, Java, Javascript, SQL, C/C++, PHP/Hack
Frameworks & Tools: Pytorch, Tensorflow, Django, Git, Vim
Data Visualization: Matplotlib, Open3D, Blender, PCL