Deep Learning for 3D Shape Modelling

Roman Klokov

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

Spécialité : Mathématiques et Informatique

Arrêté ministériel : 25 mai 2016

Présentée par

Roman KLOKOV

Thèse dirigée par **Edmond BOYER**
et codirigée par **Jakob VERBEEK**, Université Grenoble Alpes

préparée au sein du **Laboratoire Laboratoire Jean Kuntzmann**
dans **l'École Doctorale Mathématiques, Sciences et technologies de l'information, Informatique**

Deep Learning pour la Modélisation de Formes 3D

Deep Learning for 3D Shape Modelling

Thèse soutenue publiquement le **3 décembre 2021**,
devant le jury composé de :

Monsieur EDMOND BOYER
DIRECTEUR DE RECHERCHE, INRIA CENTRE GRENOBLE-RHONE-ALPES, Directeur de thèse
Monsieur JAKOB VERBEEK
INGENIEUR HDR, Facebook, Co-directeur de thèse
Monsieur VINCENT LEPETIT
PROFESSEUR, ECOLE NATIONALE DES PONTS ET CHAUSSEES, Rapporteur
Monsieur MAKS OVSJANIKOV
PROFESSEUR, ECOLE POLYTECHNIQUE, Rapporteur
Madame ANGELA DAI
PROFESSEUR ASSISTANT, Technische Universität München, Examinatrice
Monsieur RADU HORAUD
DIRECTEUR DE RECHERCHE, INRIA CENTRE GRENOBLE-RHONE-ALPES, Examinateur

Abstract

Application of deep learning to geometric 3D data poses various challenges for researchers. The complex nature of geometric 3D data allows to represent it in different forms: occupancy grids, point clouds, meshes, implicit functions, etc. Each of those representations has already spawned streams of deep neural network models, capable of processing and predicting according data samples for further use in various data recognition, generation, and modification tasks.

Modern deep learning models force researchers to make various design choices, associated with their architectures, learning algorithms and other specific aspects of chosen applications. Often, these choices are made with the help of various heuristics and best practice methods discovered through numerous costly experimental evaluations. Probabilistic modeling provides an alternative to these methods that allows to formalize machine learning tasks in a meaningful manner and develop probability-based training objectives. This thesis explores combinations of deep learning based methods and probabilistic modeling in application to geometric 3D data.

The first contribution explores how probabilistic modeling could be applied in the context of a single-view 3D shape inference task. We propose a family of probabilistic models, Probabilistic Reconstruction Networks (PRNs), which treats the task as image conditioned generation and introduces a global latent variable, encoding shape geometry information. We explore different image conditioning options, and two different training objectives based on Monte Carlo and variational approximations of the model likelihood. Parameters of every distribution are predicted by multi-layered convolutional and fully-connected neural networks from the input images. All the options in the family of models are evaluated in the single-view 3D occupancy grid inference task on synthetic shapes and according image renderings from randomized viewpoints. We show that conditioning the latent variable prior on the input images is sufficient to achieve competitive and state-of-the-art single-view 3D shape inference performance for point cloud based and voxel based metrics, respectively. We additionally demonstrate that probabilistic objective based on variational approximation of the likelihood allows the model to obtain better results compared to Monte Carlo based approximation.

The second contribution proposes a probabilistic model for 3D point cloud generation. It treats point clouds as distributions over exchangeable variables and use de Finetti's representation theorem to define a global latent variable model with conditionally independent distributions for coordinates of each point. To model these point distributions a novel type of conditional normalizing flows is proposed, based on discrete coupling of point coordinate dimensions. These flows update the coordinates of each point sample multiple times by dividing them in two groups and inferring the updates for one group of coordinates from another group and, additionally, global latent variable sample. We also extend our Discrete Point Flow Networks (DPFNs) from generation to single-view inference task by conditioning the global latent variable prior in a manner similar to PRNs from the first contribution. Resulting generative performance demonstrates that DPFNs produce sets of samples of similar quality and diversity compared to state of the art based on continuous normalizing flows, but are approximately 30

times faster both in training and sampling. Results in autoencoding and single-view inference tasks show competitive and state-of-the-art performance for Chamfer distance, F-score and earth mover's distance similarity metrics for point clouds.

Résumé

L'application des stratégies d'apprentissage profond, aux données de formes 3D pose divers défis aux chercheurs. La nature complexe de ces données 3D autorise différentes représentations, par exemples les grilles d'occupation, les nuages de points, les maillages ou les fonctions implicites. Chacune de ces représentations a vu apparaitre des familles de réseaux de neurones profonds capables de traiter et prédire en fonction d'échantillons, cela pour diverses tâches de reconnaissance, de génération et de modification de données.

Les modèles d'apprentissage profond modernes obligent les chercheurs à effectuer divers choix de conception associés à leurs architectures, aux algorithmes d'apprentissage et à d'autres aspects plus spécifiques des applications choisies. Ces choix sont souvent faits sur la base d'heuristiques, ou de manière empirique au travers de nombreuses évaluations expérimentales coûteuses. La modélisation probabiliste offre une alternative à cela et permet de formaliser les tâches d'apprentissage automatique de manière rigoureuse et de développer des objectifs d'entrainement qui reposent sur les probabilités. Cette thèse explore la combinaison de l'apprentissage profond avec la modélisation probabiliste dans le cadre applicatif des données 3D de formes géométriques.

La première contribution porte sur l'inférence d'une forme 3D à partir d'une seule vue et explore comment la modélisation probabiliste pourrait être appliquée dans ce contexte. Nous proposons pour cela un ensemble de modèles probabilistes, les réseaux de reconstruction probabilistes (PRN), qui traitent la tâche comme une génération conditionnée par l'image et introduisent une variable latente globale qui encode les informations de géométrie des formes. Nous expérimentons différents conditionnements par l'image et deux objectifs d'entraînement différents basés pour l'un sur la méthode de Monte Carlo et pour l'autre sur l'approximation variationnel de la vraisemblance du modèle. Les modèles PRN sont évalués avec l'inférence de grilles d'occupation 3D à partir d'une seule vue, sur des formes synthétiques observées à partir de points de vue aléatoires. Nous montrons que le conditionnement, par l'image observée, de la distribution a priori de la variable latente est suffisant pour obtenir des performances compétitives pour les métriques basées sur les nuages de points et état de l'art pour les métriques basées sur les voxels. Nous démontrons en outre que l'objectif probabiliste basé sur l'approximation variationnelle de la vraisemblance permet au modèle d'obtenir de meilleurs résultats que l'approximation basée sur Monte Carlo.

La deuxième contribution est un modèle probabiliste pour la génération de nuages de points 3D. Ces nuages de points sont vus comme des distributions sur des variables échangeables et utilise le théorème de Finetti pour définir un modèle global de variables latentes avec des distributions conditionnellement indépendantes pour les coordonnées de chaque point. Pour modéliser ces distributions ponctuelles, un nouveau type de flux de normalisation conditionnelle est proposé, basé sur un couplage discret des dimensions des coordonnées ponctuelles. Nous étendons également nos réseaux de flux ponctuels discrets (DPFN) de la génération à la tâche d'inférence à vue unique en conditionnant la variable latente globale a priori d'une manière

similaire aux PRN de la première contribution. Les performances génératives résultantes démontrent que les DPFN produisent des échantillons de qualité et de diversité similaires à l'état de l'art basé sur des flux de normalisation continus, mais sont environ 30 fois plus rapides que ces derniers, à la fois dans la formation et l'échantillonnage. Les résultats des tâches d'encodage automatique et d'inférence à vue unique montrent des performances compétitives et état de l'art avec les métriques de distance de chanfrein, de F-score et de distance de Wasserstein pour les nuages de points.

Acknowledgements

First of all, I want to thank my wife, who has always been my closest friend and best support during my PhD studies. This work would not have been possible without your patience and attention in all our fruitful work-related discussions and your care and cheerfulness in our everyday life.

Secondly, I want to thank my collaborators for all the discussions, provided expertise and advice you gave me, for your dedication during submission deadline periods. It was a great pleasure working with you.

Finally, I would like to thank my colleagues in the office and especially the room mates for all the relaxing breaks and some interesting and helpful talks.

About the Author

Roman Klokov completed a BSc in Applied Mathematics and Physics at the Moscow Institute of Physics and Technology and a MSc in Data Science at the Skolkovo Institute of Science and Technology before joining UGA and Inria in September 2017. His main research interests include deep learning and probabilistic modeling primarily in application to 3D data. In Skoltech he worked with professor V. Lempitsky on a point cloud recognition model and presented the results in the International Conference on Computer Vision in 2017 in a spotlight presentation. During his PhD studies Roman presented his research in probabilistic models for single-view inference and generation of 3D shapes in the British Machine Vision Conference in 2019 and European Conference on Computer Vision in 2020. The work presented orally in BMVC received a "Best Science Paper Honorable Mention" award.

List of Collaborators and Publications

The publications which constitute this thesis were produced in collaboration with Jakob Verbeek and Edmond Boyer.

Klokov, R., Verbeek, J., and Boyer, E. (2019). Probabilistic reconstruction networks for 3D shape inference from a single image. In *BMVC*

Klokov, R., Boyer, E., and Verbeek, J. (2020). Discrete point flow networks for efficient point cloud generation. In *ECCV*

Table of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

As the human world continues to rapidly move towards storing and using various kinds of information in digital formats, the demand for proper and useful instruments allowing to view, edit, and analyse digital data will only continue to increase. This is particularly relevant for visual information. The complex nature of the physical world captured by human vision was previously expressed by artists in forms of paintings and sculpture, now is almost perfectly captured and digitized by the means of photography and geometry reconstruction techniques.

Today we take for granted the tools which allow us to view and edit pictures, since almost every mobile phone has cameras and powerful tools for image editing. The same is true for geometry of 3D objects for computers. Computer software is used to create, view, and modify 3D objects for various purposes, *e.g.* 3D design of mechanical objects and their parts in engineering, reconstruction of bones and organs from CT scans and other types of non-invasive body reconstructions for medical purposes, 3D design of living space by decorators, 3D design of models and objects used in computer graphics in entertainment industry.

Together with the development of image and 3D capturing technologies, these digital kinds of human activity led to appearance of large scale databases and collections of various types of visual information. The bigger these databases grow and the more it becomes infeasible for people to perform analytical tasks on that data manually, the more obvious rises the need for automatic tools capable of solving these tasks. Modern information search engines, mobile cameras and photo enhancement software, rendering engines used in computer graphics, and lots of other technologies all benefit from recently developed powerful data-driven analytical tools based on the concept of deep neural networks.

This thesis is an addition to the development of these tools. It proposes two distinct contributions to the field of 3D shape modeling using deep learning and probabilistic modeling. The remainder of the introductory chapter is organized as follows: in Section 1.1 the general scientific context of the work is presented and used to properly motivate and place the proposed contributions within this context; Section 1.2 formulates the tasks considered in the main chapters and briefly underlines the contributions; Section 1.3 describes the overall structure of the rest of the thesis.

1.1 Context and Motivation

In order to properly formulate and motivate the contributions of this work it is essential to introduce several concepts, outlining the overall scientific context of considered tasks. To achieve that, this section briefly describes the concept of deep learning and reviews its different aspects influencing final design choices for deep neural network models (DNNs), including the choice of the final application and the variability of considered tasks, the type of the input data and desired output. Moreover, it describes the difference between visual 3D data and 2D images to properly motivate consideration of tasks defined on 3D data.

1.1.1 Deep Learning and its Applications

Due to high variability in possible designs and choices of applications, it is hard to give a comprehensive formal definition for deep neural networks and deep learning. Generally DNN is a special type of machine learning algorithm, which considers some data modality as an input, has a set of adjustable parameters (weights) and mathematical operations defined for these parameters and input data, and produces some form of an output. Pairs of adjustable weights and corresponding mathematical operations are usually called as layers of the network. Architectures of modern deep neural networks typically contain from dozens to thousands of layers, applied to inputs in a sequential manner before producing the final result. Layered pipeline of deep neural networks is not arbitrary and was inspired by the studies of brain structures. The main idea behind it is to allow the DNN model to progressively find its own understanding of the input data. Thus, after training, each layer of the network sequentially produces a new representation of the input data, which hopefully will allow to solve the final task more effectively.

The final application purpose of the model usually has the most influence on the architectures of DNNs, it defines the data modality of the required input and output and restricts the pool of suitable mathematical objectives which formalize the task. The adjustment of the model weights is called training and is achieved by stochastic optimization of these objectives called loss functions. During each training step some input data samples are fed to the network to compute the outputs and compare them with the real samples from the output data in the loss function. Then, a gradient of the loss function is computed with respect to trainable weights, which is used to update the weights to minimize the loss function. For further comprehensive introductory reading on the subject of deep learning, please refer to (Goodfellow et al., 2016).

As it was stated above, the final task for which a DNN is developed greatly influence the choices of different components of the model. Researchers have already considered vast variety of applications for DNNs and every year new challenging tasks are proposed for further study. In order to properly relate this work to concurrent deep learning research it is important to differentiate between separate categories of machine learning tasks: recognition, generation, and modification/inference tasks.

Recognition tasks are focused on allowing machines to semantically distinguish and relate considered inputs or their parts. Classification of the input 3D shapes into separate categories (*e.g.* airplanes, cars, chairs), detection in images returning tight bounds of the desired objects (*e.g.* face detection in digital cameras), and search of semantically relevant data with respect to a query in search engines, are all examples of the recognition tasks. The main purpose of developing solutions to these tasks is to allow automation of data understanding. As mentioned, it is essential for modern public and private organizations to have large scale data

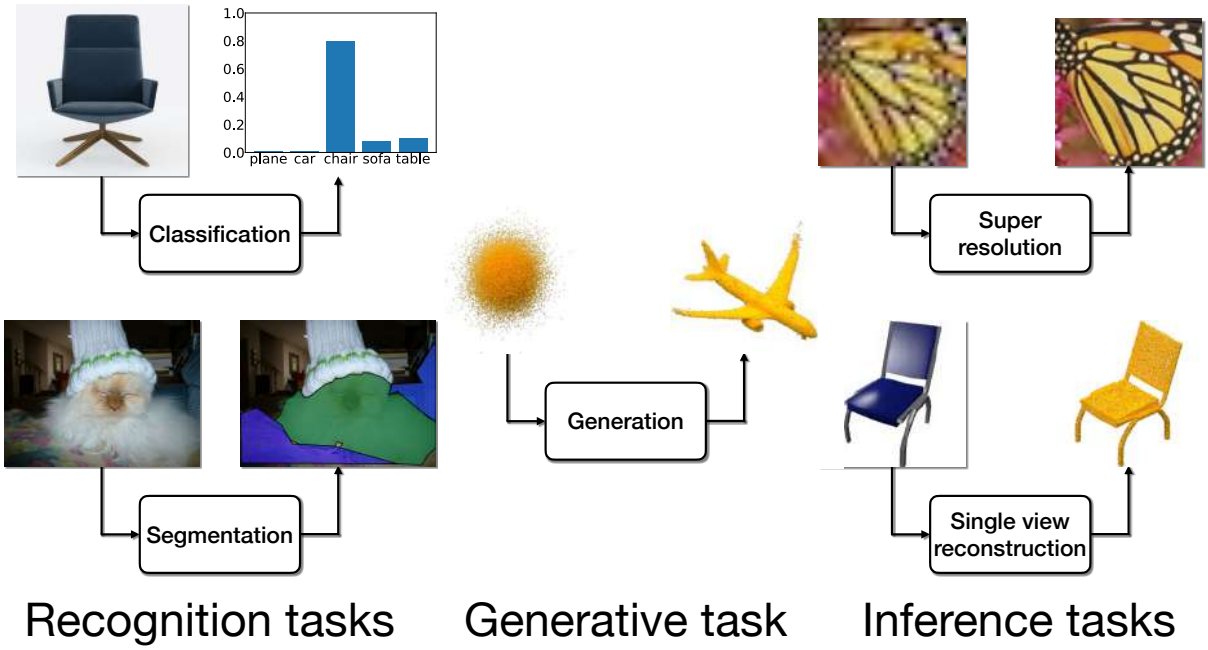

Figure 1.1: Examples of machine learning tasks with their inputs and outputs. Left column show 3D shape classification and semantic segmentation of images as examples of recognition tasks. Central column considers point cloud generation from input noise as an examples of generative tasks. For modification and inference tasks we show image super resolution and 3D point cloud reconstruction from a single image in the right column.

processing automation tools in an era of digital data if they want to efficiently analyze it and avoid tedious and often infeasible manual labour.

Generative tasks do not just aim at extracting features, relevant for semantic understanding of the data, they pose a more challenging problem of realistic imitation or, in other words, modeling of data. Rather than learning some categorization or relevance degrees, they learn to produce synthetic data samples. In a broader sense, it is a generalization of the recognition tasks, since in that case, models do not just extract information to make predictions about distinct data features, but rather try to completely capture all the data features to be able to reproduce them. This idea is supported by a nowadays standard experiment in the literature (*e.g.* for 3D shape classification refer to (Girdhar et al., 2016)), which shows that the features learned by generative models directly from data samples without any semantic annotations could be successfully used to solve recognition tasks and be on par with the features from recognition models which used semantic annotations during training. This, in particular, is a strong argument in favour of further research and improvement of generative models, since data annotation is generally expensive and is not always a feasible option on a large scale.

Content generation is still one of the most challenging tasks scientifically but is not the only possible and presumably even not the most promising application for generative models at the moment. Besides providing discriminative features suitable for recognition tasks, generative data modeling is closely tied to data modification or data inference tasks. These tasks are very close and generally consider two different modalities of the same data as inputs and outputs of the model. The difference between them lies in, what can be called, the distance between input and output modalities. For example, noisy and clean images in the image denoising

task are both two different domains of the same image modality and are generally close, while an input image and output 3D shape in a single image 3D shape reconstruction task both represent the same object, but come from different and possibly not very well connected data modalities.

This set of tasks is closely linked to generation, because they share the type of the desired model output: plausible data samples. This results in the use of the same network components for prediction. Moreover, these tasks can be viewed as conditional generation of target data samples given samples from the input data, which brings them even closer and allow for a full transfer of techniques from one category of tasks to another. As the result, both data modification tasks (image denoising, colorization, super-resolution, *etc*.) and data inference tasks (image and 3D shape completion, 3D shape reconstruction from a single image, text to image generation, *etc*.) can benefit greatly from advances in generative modeling. Other works on interpretability of modern generative models (Bau et al., 2020) propose a framework, that allow to realistically modify parts of images using semantic labels, *e.g*. change the shape of the windows on the house from a square to a circle. These advances in data modification tasks which are achieved by the use of generative models and their components are leading to their future use in the automation of the various content design processes and tools. We demonstrate several examples of tasks considered by modern DNNs in Figure 1.1.

1.1.2 Data Influence on Design Choices in DNNs

Input and output data modalities have a direct effect on the architectures of DNNs. In fact, most of essential types of layers present in modern architectures were developed specifically for particular data types. 2D/3D convolutional layers for images/voxelized volumes, graph convolutions for graph data, shared fully-connected layers for sets of points, *etc*. all rely on distinctive intrinsic features associated with their target input data types. Convolutions for 2D/3D inputs rely on regular grid structures of images and voxel occupancy grids, as well as on the presence of the local correlations (patters) in such data. Graph data is not regular, so a non-trivial adaptation of regular convolutions that uses graph spectral analysis was developed. Video data is an instance of ordered sets of images data type. The sequential nature of such data inclines to the appearance of the temporal contextual modules such as long short-term memory (LSTM) modules (Ng et al., 2015), which are capable to progressively update extracted features given next frames. All these examples demonstrate that the type of the input data and its associated features restricts some mathematical operations and promotes others.

This particular influence of the input choice is further amplified by the existence of different representations of the same data. A perfect case to show it is the 3D shape classification task. In that task models are trained to predict semantic labels for any input 3D shape. The nuance is that, in fact, 3D shapes can be numerically represented in different ways with the help of various data types: voxel occupancy grids, point clouds, meshes or graphs, multi-view image sets, *etc*. While the task stays the same for all these options, the architectures of the DNNs suitable for each representation are drastically different, and it is not clear beforehand if some particular combination of input representation and a DNN designed for it will be superior.

Finally, the application purpose usually dictates particular type of output from a DNN. The same task can be solved differently, so the output data format depends on the mathematical formulation of the task and the chosen loss function. In generative and inference tasks the choice of the output data type is even less definite. Since in that case plausible data samples

are the desired output of a model, similar architectural variations occur compared to options allowed by differences in the possible model inputs. Most data types require careful design of appropriate DNN components capable of outputting them. Unlike recognition models that are designed to output low-dimensional (compared to inputs) vectors of features or probabilities, generative models need to output high-dimensional data samples that preferably preserve sharp details. The variability of the output representations, combined with the desire to preserve sharp features in generated samples, resulted in the appearance of rich families of DNN architectures corresponding to each data type.

1.1.3 3D Data

The initial success of convolutional neural networks applied to large scale image datasets (LeCun et al., 1998; Krizhevsky et al., 2012) ignited a live interest in application of deep learning methods to other data domains, including 3D data. 3D data is a complex data type which, in fact, combines several modalities: geometry, appearance, and shading. Geometry captures the spatial configuration of surfaces of shapes or scenes and can be represented in different formats, *e.g.* points, sets of polygons, implicit functions. Appearance provides corresponding colors at every point of surfaces by the means of textures represented as images, and texture coordinates which define for every geometric primitive where the related color information is situated in the corresponding textures. Shading information is the intensity of the color at a given surface point associated with the geometry of shapes (normal to the surface of the object at a given point), their physical properties (for example reflectance), and light sources configuration present in the scene. The combination of geometry, appearance and shading data modalities contains sufficient information to render images of 3D objects and scenes from arbitrary viewpoints. In that sense, visual 3D data is a generalization of commonly used 2D image data, which is always partial because of the often present uncertainties, associated with object cropping, self-occlusions, and shading issues (overly dark or bright scenes). Therefore, in order to reach comprehensive autonomous machine understanding of visual information it is required to consider geometry and illumination present in scenes.

Machine understanding of the geometric component of 3D data is a unique task which plays a crucial part in a variety of applications. Automatized understanding of the spatial configurations and dimensions of objects and scenes is essential for effective simultaneous localization and mapping (SLAM) and navigation solutions used in robotics and self-driving cars. Generation of geometry samples could potentially improve 3D content creation tools used by designers and engineers. The advances in inference tasks could bring direct improvement to 3D shape reconstruction, completion and other related applications.

However, application of DNN methods to the geometric part of 3D data, especially in generative/inference tasks, poses numerous challenges among which the most striking are:

- the choice of the representation for 3D geometry, and the influence of this choice on architectures of DNNs;
- the choice of suitable metrics and evaluations protocols;
- the choice of mathematical formulation of considered tasks providing a final loss function;
- problems with existing synthetic data and lack of real world datasets.

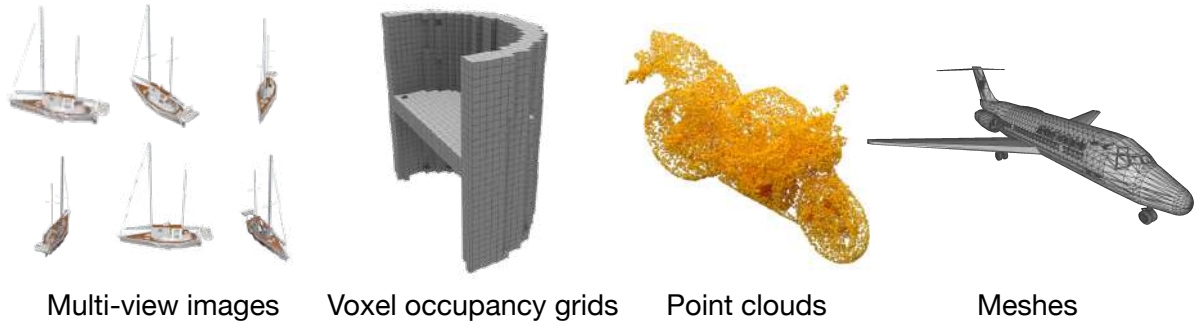

Figure 1.2: Examples of 3D shape geometries represented by various data types. From left to right: sets of images taken from different viewpoints; 3D regular occupancy grids indicating the interior of a shape; set of points randomly sampled from the surface of a shape; collection of polygons, approximating the surface of a shape in a piece-wise manner.

3D Shape Representations

Geometry of 3D shapes can be represented in vastly different ways. Classical reconstruction approaches and some recognition DNNs use sets of images or renderings as an input representation of 3D shapes. Most synthetic datasets of 3D geometries are represented with meshes: collections of polygons (often triangles) sharing vertices and edges that approximate shape surface with planes in a piece-wise manner. Datasets, coming from distance measuring equipment, *e.g*. LiDAR, naturally use point cloud representation. It assumes, that every point in a cloud is situated on surfaces of objects in the scene, and assigns coordinates to it. Contrary to the previously listed options, the voxel grid representation considers regular discrete partitions of volumes containing shapes and uses binary occupancy for every cell of these grids to define if it is inside or outside of the object. Implicit function representation is a generalization of voxel grid representation for continuous locations. It considers continuous coordinate values as inputs and returns either an indicator value showing if this point is outside or inside the object, or the signed shortest distance to the object surface. Zero level set of this function represents the surface of an object. See Figure 1.2 for visualizations of 3D shapes using different representations.

All these options use different underlying data types. Images and voxel grids are regular structures defined over the whole volume and its 2D projections, point clouds are unordered sets, meshes are graph data, implicit function is a parametrization of a surface with a function. In the absence of universal approaches capable of working with any option, and with the desire to improve existing solutions, researchers spawned separate streams of works devoted to designing appropriate DNNs capable of extracting features from and outputting each of those representation. These related works will be reviewed in detail in the next chapter.

Evaluation of 3D Shape DNNs

Evaluation for recognition models working with 3D data is standard and similar to evaluation methods used for 2D image data models, since the choice of input representation does not usually restrict the use of final recognition metrics. However, the case is entirely different for 3D geometry inference and generative tasks, where the goal is to evaluate similarity between ground truth and generated or reconstructed samples. In case of inference or reconstruction, where ground truth is available during evaluation, problems mostly stem from differences in

representations. Since representations use different data types, they have to rely of differently defined similarity measures.

For voxel grids similarity is measured with intersection-over-union metric, proposed for this context in the work of (Choy et al., 2016):

$$\text{IoU} = \frac{\sum_c S_c^p \wedge S_c^t}{\sum_c S_c^p \vee S_c^t}, \tag{1.1}$$

where S^p and S^t are predicted and ground truth binary occupancy grids, c is a voxel grid cell index, $\wedge$ is a logical and, $\vee$ is a logical or operator. It considers a ratio of the number of occupied cells in the intersection of the predicted and ground truth grids to the number of occupied cells in the union of these grids. While this metric is natural and convenient for the voxel grid representation it is highly sensitive to the resolution of the grid.

Results obtained with other representations, however, are mostly evaluated using point clouds sampled from them. Point clouds are sampled uniformly from meshes, while implicit functions firstly are converted to meshes with the help of dense occupancy probing in regular volumetric grids and application of the marching cubes algorithm (Lewiner et al., 2003). It is also possible to obtain point clouds from volumetric occupancy predictions by the application of the same marching cubes algorithm. Several similarity metrics were proposed to evaluate resulting point clouds sampled or reconstructed from DNN models, including Chamfer's distance and earth mover's distance first used in the pioneering 3D shape reconstruction from a single image paper (Fan et al., 2017), as well as an F-score, proposed for reconstruction by (Knapitsch et al., 2017).

Chamfer's distance is defined as follows:

$$\text{CD} = \sum_{x \in S^p} \min_{y \in S^t} \|x - y\|_2^2 + \sum_{y \in S^t} \min_{x \in S^p} \|x - y\|_2^2, \tag{1.2}$$

where x and y are points in predicted and ground truth point clouds S^p and S^t. For each point in one set it finds the closest point in another set and sums obtained squared distances, the same is done symmetrically for points in the opposite set. Some works *e.g.* (Sun et al., 2018) studied how the human perception of output point clouds correlates with different evaluation metrics for point clouds and advocated for the use of Chamfer's distance since it showed best results. Others (Tatarchenko et al., 2019) demonstrated that the metric is sensitive to outliers. From the definition it is clear that a) several points from one set can be matched to a single point from another set, b) the metric do not take into account varying local densities in both sets. As the result, CD is not sensitive to sampling irregularities, as demonstrated in experiments performed by (Wang et al., 2020). Despite these problems, it remains one of the most used evaluation metrics for DNN models working with 3D geometry data.

The earth mover's distance is defined for equally sized point clouds as:

$$\text{EMD} = \min_{\phi: S^p \to S^t} \sum_{x \in S^p} \|x - \phi(x)\|_2, \tag{1.3}$$

where $\phi(\cdot)$ is a bijection. In principle, EMD should assign pairs of points from different sets in an optimal way by solving optimization problem. In practice, however, exact solutions are computationally prohibitive and, approximations are used instead. Unlike CD, EMD establishes unique per point correspondences between points in predicted and ground truth sets, which means it is sensitive to sampling irregularities and differences between the considered sets,

which was studied by (Wang et al., 2020). Introduced approximations, however, inject noise in the metric values, since they do not necessarily produce the same bijections in different runs.

The F-score is defined in a classical way as a harmonic mean of precision and recall:

$$F(\tau) = \frac{2P(\tau)R(\tau)}{P(\tau) + R(\tau)}, \tag{1.4}$$

where precision and recall are calculated as functions of threshold τ:

$$P(\tau) = \frac{100}{|S^p|} \sum_{x \in S^p} \left[\min_{y \in S^t} \|x - y\| < \tau \right], \tag{1.5}$$

$$R(\tau) = \frac{100}{|S^t|} \sum_{y \in S^t} \left[\min_{x \in S^p} \|x - y\| < \tau \right], \tag{1.6}$$

where $|\cdot|$ denotes the cardinality of point sets, and $[\cdot]$ is the Iverson bracket. Thus, precision and recall counts numbers of points in one set that lie within threshold distance from points in other set. By construction, this metric is sensitive to both quality of reconstructed details and completeness of the reconstruction, however it requires a proper choice of the threshold value since some values can be over permissive or restrictive.

Since every representation can be used to obtain point cloud predictions it is tempting to establish a unified evaluation protocol and compare them through their proxy point cloud results. This approach, however, has underlying issues. In principle, each 3D geometry representation is an approximation of exact true surfaces of shapes. Since these approximations are usually done with different levels of granularity for each representation, their direct comparison may not be strictly fair, unless it is verified that every compared option approximates ground truth with similar granularity. Moreover, it is possible, that learned representations may perform differently depending on the chosen granularity of the ground truth shape approximation, making some of them suitable for crude and others for precise reconstructions. Besides the precision, the scalability of representations and according methods also varies, so, given particular memory or computational limitations, some representations may be favourable over the others.

Additional difficulties in evaluation rise for generative models. A general problem for generative models outputting any data is the absence of corresponding ground truth per every generated sample. To overcome this, researchers (Lopez-Paz and Oquab, 2017; Achlioptas et al., 2018) proposed to use aggregate metrics calculated over entire sets of ground truth and generated data samples, evaluating both diversity of the generated set and similarity between these sets. These will be discussed further in Chapter 4.

Geometry Learning Approaches

The choice of the suitable learning method providing the loss function for model parameters optimization is one of the principal decisions to be made when designing a DNN model for generation and inference tasks. Several conceptually different approaches were developed or adapted from DNNs working with other data domains in application to 3D geometry data:

- optimization of a chosen similarity metric between generated/inferred and ground truth samples;
- likelihood maximization in a probabilistic model of target data distribution;

- learning a rich support set of the target data distribution in an adversarial manner.

The most straightforward way to enable training in autoencoding and inference models is to directly optimize one of the similarity metrics used for evaluation as a loss function. In that case, it is required that the chosen metric can be efficiently differentiated through to pass gradients during backpropagation. This is not always the case for all the metrics mentioned earlier. IoU and F-score are not differentiable, EMD is calculated with the help of approximations so the same holds for its gradients, only CD can be differentiated through efficiently. Naturally, such an approach inherits the drawbacks of the optimized metrics. For example, in case of autoencoding models for point clouds optimized with CD, learned shapes often tend to concentrate produced points near dense regions of ground truth rather then spreading them evenly over the surface. Moreover, in case of metric optimization, resulting models are susceptible to overfitting to chosen metrics to the detriment of others.

Another approach involves the use of probabilistic modeling that strictly defines predicted data distributions and optimize the likelihood of ground truth data samples calculated within these models. Various adaptations of variational autoencoders, firstly introduced for images by (Kingma and Welling, 2014), are one of the most commonly used approaches involving probabilistic modeling for generative tasks. These adaptations model target data distribution with the help of additional low-dimensional latent variable. The posterior distribution of the latent variable encodes global information about particular data samples and is regularized to be close to unconditional prior distribution of the latent variable. Inference in the model is enabled by the variational approximation of the model likelihood providing the loss function, and reparametrization trick enabling sampling from the latent variable posterior in a differentiable manner. After training, samples from the latent variable prior can be decoded into plausible data samples. The same approach can be successfully applied to inference tasks, which will be discussed in Chapter 3.

Since this approach avoids direct metric optimization, models trained by it avoid drawbacks associated with the optimized metrics. Unfortunately, it has its own difficulties, which include: 1) the necessity to introduce probabilistic models for each representation, which is rarely trivial due to the fact that each representation use a unique data type; 2) inference in such models is often non-trivial and requires different approximations instead of exact computations; 3) data distributions of various 3D geometries have complex nature, are problematic to model and usually require long training times to be captured by DNNs.

Generative adversarial networks were also originally proposed for image data by (Goodfellow et al., 2014) and introduced a special training algorithm for generative models based on the competition between two networks. According to this method, generator network learns to produce data samples from input noise samples, while discriminator network takes both ground truth and generated data samples and is trained to distinguish them by the means of binary classification. Thus, during training, these networks compete with each other: the generator learns to produce samples indistinguishable from real ones and the discriminator learns to find even the slightest differences between them to classify them properly. Similarly to probabilistic modeling, this approach also avoids direct specified metric optimization and instead relies on an implicit similarity through the real/fake classification. Unlike probabilistic solutions, it avoids explicit modeling of target data distributions and learns to produce a support set of this distribution. In practice, it often leads to generation of more realistic samples containing more sharp features. However, adversarial approaches are notably hard to setup for effective training in terms of tuning various model hyperparameters. Even the slightest changes in the

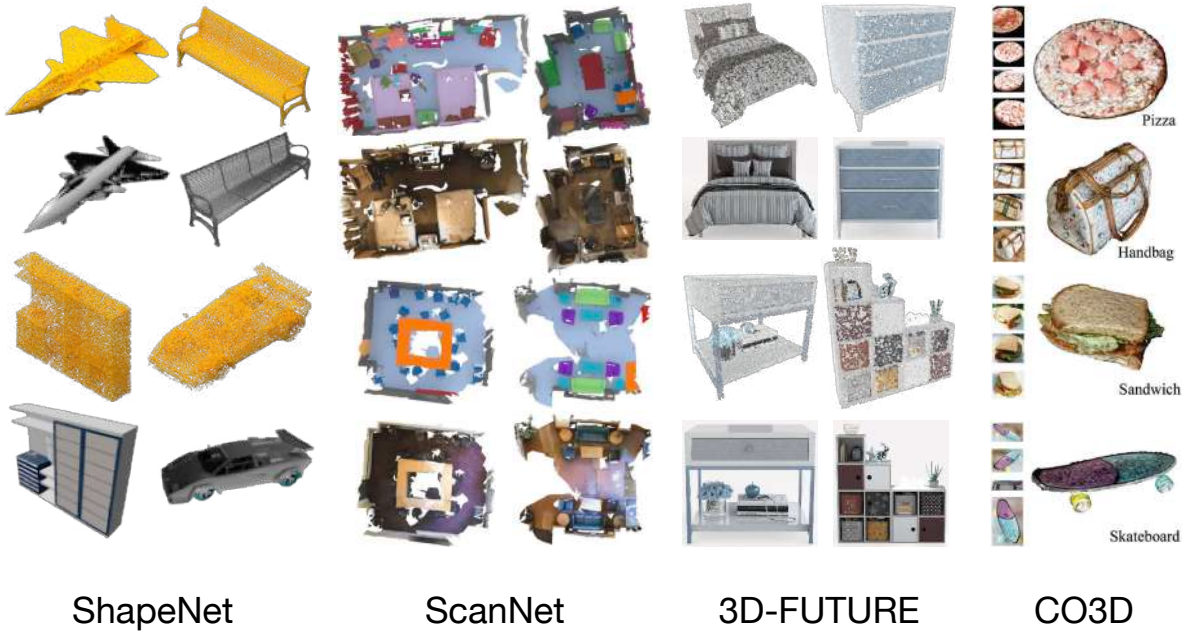

Figure 1.3: Samples from common and recent 3D datasets. From left to right: point cloud and rendered image samples from ShapeNet (Chang et al., 2015); ground truth meshes and scans from ScanNet (Dai et al., 2017); colored point clouds and image renderings from 3D-FUTURE (Fu et al., 2020); frames from ground truth videos and corresponding ground truth point clouds from CO3D (Reizenstein et al., 2021)
.

setup could lead to degenerate model behavior, known as mode collapse. In mode collapse, the model learns to produce a small support set of very similar samples that can efficiently fool the discriminator, even if they do not look realistic in terms of human standards.

We provide both qualitative and quantitative comparison of 3D shape autoencoding reconstructions produced by various generative models based on all the principles described above in Chapter 4.

3D Datasets for Geometry Learning

To this day, a limited number of synthetic and real datasets are used for 3D geometry learning in generative and inference tasks. The most popular synthetic datasets are ModelNet (Wu et al., 2015) and ShapeNet (Chang et al., 2015). Both datasets are collections of 3D shapes with textures, represented as meshes and labeled into semantic categories: cars, planes, chairs, tables, *etc*. It is hard to overstate the importance of these datasets for current progress in both recognition and generative/inference tasks. Nowadays they are used as a standard for evaluation of most of the new models. However, samples are often not ideal in these datasets for geometry learning purposes. There is a significant number of non-watertight shapes, which have errors or missing parts in their mesh topology resulting in holes. These erroneous samples create confusion for models during training resulting in fuzzy predictions in the most uncertain regions. In case of single class subsets, there are extreme variations in the level of the details captured by the ground truth meshes, *e.g*. for cars category there are samples that only have a mesh of the outer car surface and samples with meshes capturing the surfaces of the interior

of the car or the interior under the car's hood. It is possible to argue that these aspects of these datasets might be even beneficial for recognition models since the recognition results are usually confident and this variance provides additional challenges for models. However, such extreme variance in data samples is not ideal for generative and reconstruction models since including it means that models will learn to produce erroneous samples which contradicts the intended behaviour of such models.

Other datasets of synthetic shapes, including Thingi10K (Zhou and Jacobson, 2016), and ABC (Koch et al., 2019) provide collections of 3D shapes gathered from existing online platforms dedicated to sharing 3D model designs of various objects suitable for 3D printing and other applications. Although both datasets are also not perfect in terms of containing only clean shapes without various errors, the authors evaluated geometric quality characteristics and ensured that the data in these datasets is cleaner compared to ShapeNet. The increased quality and semantic variability of samples make them better candidates for pure geometry learning tasks, especially surface reconstruction from input point clouds. On the other hand, this variability and the lack of distinct categorisation of samples into classes compared to ShapeNet make these datasets extremely challenging for generative models. Even though the data quality is better, to our knowledge, there are still no evaluations of generative models on these datasets. Recently, a new large dataset of synthetic furniture shapes rendered in various realistic room interiors was released (Fu et al., 2020). Although both the shapes and the backgrounds used for rendering are synthetic, the advantage of this dataset is the curated nature of data, since it was provided by professional designers.

The choice among the real 3D shape datasets is limited at the moment. Pix3D dataset (Sun et al., 2018) is a collection of CAD models of different furniture, which are paired with collections of real images of according furniture pieces in realistic environments. Such data can be used for realistic evaluations of 3D shape inference from a single image task, unlike synthetic shape dataset for which the images are produced by rendering shapes and using either random or synthetic backgrounds. Unfortunately though, there are not enough samples to use this dataset for training. Other real datasets, like ScanNet (Dai et al., 2017) provide RGB-D scans of real room environments with according reconstructions and per-instance semantic annotations. Compared to synthetic environments and shapes such data contains features specific to the real data acquisition process, such as non-uniform density regions, holes, noise and other artifacts. Thus, such data is better suited for models developed for real data applications. Another type of the real data suitable for 3D deep learning is the motion scans of real humans, *e.g*. FAUST dataset (Bogo et al., 2014). Although the overall geometry variance in such data is limited to representing different human body compositions and poses, it provides other challenges, associated with the non-rigid nature of human body geometry and the temporal component, available in that data. These aspects make this data particularly suitable for the task of dense correspondence prediction between different time states of the same or different people.

Recently, another collection of real 3D data was proposed by (Reizenstein et al., 2021). The dataset consists of the videos of approximately 19K objects from 50 MS-COCO (Lin et al., 2014) categories. Every video was put through multi-view dense point cloud reconstruction pipeline to obtain corresponding ground truth 3D data. The quality and the increased size of this dataset make it a promising option for future research in various DNNs applications for 3D data.

Researchers often tend to focus on the importance of the real data evaluations for all the freshly proposed 3D data DNN models and judge the absence of such experiments negatively.

Due to this, it is important to underline that all these datasets better suit different applications. For example, if the target purpose of the designed 3D data model is the query similarity search in a database of CAD models it does not make sense to train and evaluate such model on real scans containing various artifacts, so CAD models should be used instead. On the other hand, if the purpose of the model is to provide surface predictions from input noisy point clouds, real scans would be more relevant for that purpose. Both recognition and generative/reconstruction tasks could be relevant for both synthetic and real data regardless of the other type of data, thus new improved feature extraction and data sample generation/reconstruction DNNs are always valid regardless of the type of data they were evaluated on.

1.2 Contributions

This work builds upon earlier work involving the development of DNN models in application to 3D data. It mainly considers generative and inference applications, with a particular focus on the development of principled probabilistic variants of models suitable for these applications. It is based on two distinct contributions:

1. The first contribution considers the task of 3D shape inference from a single image. It proposes to use probabilistic modeling framework and describes a family of latent variable probabilistic models, Probabilistic Reconstruction Networks (PRNs), which formalizes the task of learning the distribution of the ground truth 3D shapes conditioned on input images. PRNs compare various possible image conditioning options in different components of the model, *e.g.* in the prior and posterior distributions of the latent variable, in the 3D shape decoder. Two different likelihood approximations are considered for maximization to train models. Additionally, a specific variant in the proposed family of models links the single-view 3D shape inference task with the generation task and allows joint training for both of these tasks with a single model and objective. Moreover, the proposed formalization of the task allowed to clearly categorize existing approaches in terms of how they treat such aspects of the task as image conditioning mechanism and training procedures.
   In practice, we implement the resulting family of models with deep convolutional neural networks for voxel grid 3D shape representation and evaluate them on the subset of the ShapeNet dataset designed for single and multi-view reconstruction introduced in the work of (Choy et al., 2016). We show that conditioning the latent variable prior on the input images is sufficient to achieve competitive and state-of-the-art single-view 3D shape inference performance for point cloud based and voxel based metrics, respectively. We additionally demonstrate that the probabilistic objective based on variational approximation of the likelihood allows the model to obtain better results compared to Monte Carlo based approximation and either of the deterministic counterparts.
   This work was published and presented orally in the British Machine Vision Conference in 2019, where it received the "Best Science Paper Honourable Mention" award. The implementation of PRNs was made public and is available online[1].

2. The second contribution presents a novel generative model for 3D point clouds. This work applies probabilistic modeling to the point cloud generation task and proposes a latent variable model which treats each point cloud as a distribution of points on the

[1] https://github.com/Regenerator/prns

object's surface. In contrast to most related work, the model allows to model point clouds of arbitrary sizes and makes use of a novel variant of conditional coupling normalizing flows designed to model point cloud data distributions given latent variable samples as inputs. The invertible nature of normalizing flows allows for efficient calculation of conditional likelihood. Additionally, we propose an extension to our Discrete Point Flow Networks (DPFNs) that allows application to the single-view shape inference task.
Evaluations on point clouds sampled from ShapeNet CAD models are performed for generation, autoencoding and single-view inference tasks. Generative results show that DPFNs are approximately 30 times faster both in training and inference compared to a similar model based on continuous normalizing flows, PointFlow (Yang et al., 2019), while being able to obtain similar or better results in terms of quality and diversity of generated samples. Autoencoding and single-view inference evaluations demonstrate state-of-the-art and competitive performance in terms of the EMD and other point cloud similarity metrics respectively. Additionally, two different ground truth data normalization and scaling options are compared to show that per-object normalization and scaling results in improved performance.
The work was published in the European Conference on Computer Vision in 2020. The code for DPFNs is available online[2].

1.3 Thesis Structure

The remainder of this thesis is organised as follows:

- Chapter 2 contains an overview of existing DNN methods developed for 3D data processing. They are categorised into three groups, depending on the type of tasks they consider: recognition, generative and inference tasks.
- Chapter 3 presents the first contribution of the thesis, which includes the probabilistic framework for single-view 3D shape inference task, its implementation by the means of DNNs called Probabilistic Reconstruction Networks, and experimental results obtained with them for single-view 3D shape inference task.
- Chapter 4 presents the second contribution, which includes the probabilistic model for 3D point cloud generation, the architecture of conditional coupling flows suitable for point clouds, DNN implementation of the model called Discrete Point Flow Networks, and experimental results obtained with the model in generation, autoencoding and single-view 3D shape inference tasks.
- Chapter 5 underlines the importance of the presented contributions and discusses possibilities for further research in related tasks.

[2] https://github.com/Regenerator/dpf-nets

CHAPTER 2

Deep Learning for 3D Shape Processing

The following chapter aims to give a comprehensive overview of existing deep learning methods applied specifically to various modalities of 3D data. It categorises related work according to the type of the final application: recognition, generation, modification/inference, and reviews numerous important works in each section.

2.1 Recognition Models

Prior to deep methods numerous global and local 3D shape descriptors were proposed, based on rendered views, probability distributions, geometric properties of shape surfaces and many more. For a comprehensive review of these methods consider reading the work of (Kazmi et al., 2013).

Recognition models played a significant role in the initial rise of the deep learning methods applied to 3D data. Since they mostly focus on the feature extraction aspect of the task, it is natural that deep recognition approaches introduced first neural network architectures suitable for processing of different input 3D geometry representations into corresponding low-dimensional feature vectors. These network and layer architectures now could be considered as the most basic and essential building blocks for all the other models, applications, and input representations, since feature extraction is required for almost any imaginable task. Moreover, these architectures often gave inspiration or directly provided applicable network components to the models designed for autoencoding, generative and inference tasks.

Since this thesis is mostly centered around generative and reconstruction applications, in this section we review selected existing works categorized according to the type of the encoded representations and give preference to approaches that inspired corresponding decoding components.

2.1.1 Recognition of Occupancy Grids

3D ShapeNets (Wu et al., 2015) was one of the first deep learning models for 3D data in general and voxel grids in particular. It proposed to use voxel grids of occupancy probabilities to represent 3D shapes for recognition and generative tasks and to apply 3D convolutional layers to extract features from such inputs. Complete architecture was based on the idea of Deep Belief Networks (Hinton et al., 2006) and was encumbered by its per-layer training algorithm.

VoxNet developed by (Maturana and Scherer, 2015), used similar 3D CNN architectures for feature extraction but used a now standard end-to-end training approach: stochastic gradient descend paired with backpropagation for proper simultaneous calculations of gradients with respect to each layer's weights. The classification model of (Brock et al., 2016) significantly increased the complexity of the baseline 3D CNN architecture by introduction of multiple network components and design features which have already proven their effectiveness for 2D CNNs including batch normalization (Ioffe and Szegedy, 2015), residual connections (He et al., 2016a) and inception-based modules (Szegedy et al., 2017). Resulting combined architecture and especially the ensemble of such architectures provided performance that is still close to present state of the art.

The regular structure of voxel grids allows to directly transfer suitable techniques from more popular and, thus, further developed 2D CNNs designed for images. However, since voxel occupancy grids and 3D convolutions scale cubically with respect to the grid resolution, this quickly becomes problematic for higher resolutions. To improve this OctNet (Riegler et al., 2017) exploits the inherent sparsity of the voxel grid representation and proposes a custom data structure for voxel grid based on shallow octrees. Further, the authors formulate principal neural network layers, required to compose a 3D convolutional DNN, in a way compatible with this data structure. Resulting OctNet not only demonstrates state-of-the-art performance among voxel-based models but also is capable of processing high resolution grids while being faster and more memory efficient compared to standard 3D convolutional DNNs.

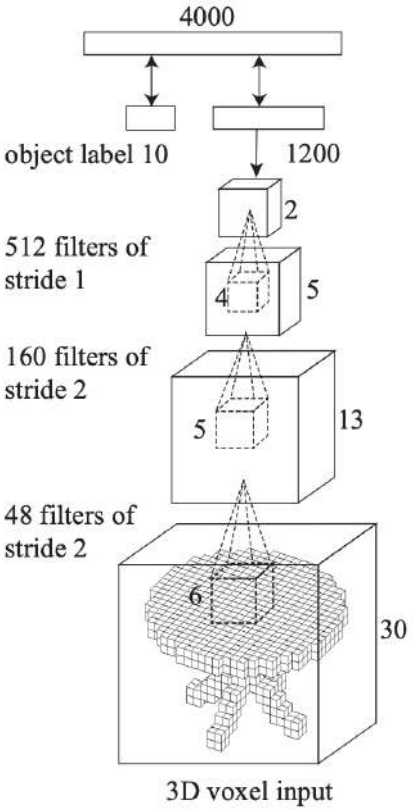

Figure 2.1: Architecture of the 3D ShapeNets CNN for shape recognition. Visualization is taken from (Wu et al., 2015).

2.1.2 Recognition of Point Clouds

Because of the issue of the high computational complexity of voxel-based 3D data models, researchers explored other more scalable representations. Point clouds are one of the most natural options for such a substitute, since they are commonly used, and allow for processing methods with linear scalability with respect to the number of input points. To achieve that, however, a special type of neural network architectures had to be developed. These architectures should treat point clouds as unordered sets of points and propose according neural network layers capable of being expressive enough to solve desired tasks but, at the same time, treating all the points equally in a mathematical sense, since the results should be invariant to permutation of input points in a cloud.

These properties desired for DNNs working with point clouds were achieved for the first time in the PointNet architecture (Qi et al., 2017a). To produce features PointNet uses fully-connected layers shared for all the points in a cloud. Thus, every per-point feature vector is produced in an independent manner. Additionally, PointNet proposes to predict extra transformations for input and intermediate point features by small subnetworks conditioned on the transformed

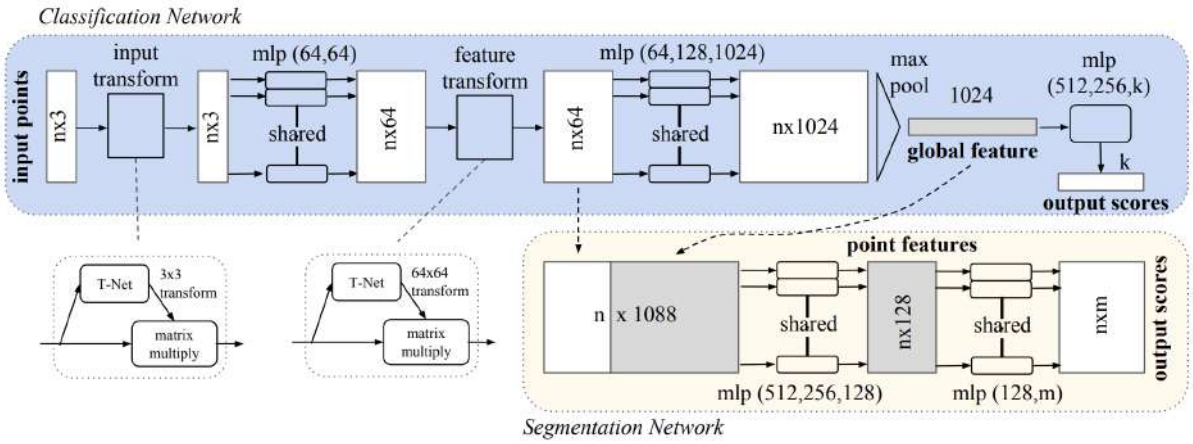

Figure 2.2: Architectures of PointNet for classification and semantic segmentation tasks. Shared MLPs are used to extract per-point features which are aggregated to obtain global features by a max pooling operation over all the points. Resulting vector is used for classification or concatenated with individual point features for segmentation. Picture is taken from (Qi et al., 2017a).

target inputs or features. The authors argue that such a transformation aligns data and allows to achieve invariance to transformations of input point clouds. To produce global features PointNet uses maximum or average feature pooling over all the individual points. For classification these global features are fed to an MLP to produce class probabilities; for semantic segmentation global features are duplicated and concatenated with individual point features prior to prediction of the shape part class probabilities. Resulting PointNet produces results surpassing the performance of the most of voxel-based methods. A more principled formulation of NN architectures, supporting the processing of point clouds is described in the work of (Zaheer et al., 2017).

Original PointNet do not have any inherent hierarchical feature aggregation mechanisms, which could be restrictive for better shape parts understanding. To tackle this problem several works propose different improvements over PointNet that make use of various data structures over point clouds, introducing hierarchical partitions over them. The approach of (Klokov and Lempitsky, 2017), improved in (Yi et al., 2017), proposes 1) to use binary spatial partition trees, namely kd-trees and pd-trees, to precompute a hierarchy of points; 2) to share parameters of MLPs only among the points corresponding to a particular configuration of the current tree node; 3) to hierarchically pool point features corresponding to the children of a particular tree node. Thus, instead of the per-point features, Kd-Networks produce features per every tree node. Alternatively, a direct continuation of PointNet (Qi et al., 2017b) uses prefixed radii ball query trees to build corresponding hierarchies for feature calculation and aggregation.

The irregular nature of point clouds prevent models from direct use of standard convolutional layers. Nevertheless, special type of convolutions are proposed specifically for point clouds in the work of (Li et al., 2018b). Similarly to hierarchical feature aggregation approaches, PointCNNs define a nearest neighbour structure based around downsampled subclouds, calculate features and combine them, in this case, by standard 1D convolution. Standard convolutions are sensitive to point permutations, so, trainable feature projections are applied before them to enable partial robustness to possible permutations.

Dynamic Graph CNNs (Wang et al., 2019b), again, uses a tree of kNN structures to define

aggregation graphs, but define graph convolution with symmetric aggregation operation to inherently obtain invariance to point permutations. DGCNNs also recompute the nearest neighbour graph for further feature calculations directly using intermediate features, contrary to complete precomputation of full trees performed in the prior work.

Recently, the authors of (Liu et al., 2019) argued that both voxel and point cloud representations are not ideal for DNN processing because of the cubic scaling with increasing grid resolutions and discontinuous storing in memory of neighbouring points. To overcome this, they propose point-voxel convolution which combines methods for both representations by using MLPs to extract per-point features, similarly to PointNet, but, at the same time, to convert point clouds in low-dimensional grids, extract 3D convolutional features from them for improved locality (compared to other point feature aggregation approaches), and to extract per-point features from resulting grids with trilinear interpolation. The authors show, that such a combination is effective in terms of being competitive with state-of-the-art recognition methods and being more efficient in terms of required computations.

2.1.3 Recognition of Meshes

Initial methods for mesh recognition proposed generalization of regular convolutions to graph data. The first work on graph convolutions (Bruna et al., 2013) proposes two possible constructions for them: one based on hierarchical clustering of graphs in the graph feature space, and another based on moving to the spectral domain of graphs and defining convolutions similarly to the spectral interpretation of standard convolutions. This work shows that that graph CNNs defined in both feature and spectral domains could be similarly effective. A series of follow-up works (Henaff et al., 2015; Defferrard et al., 2016; Kipf and Welling, 2016) based on the idea of performing graph convolutions in the spectral domain consider different forms of filters in spectral convolutions for improved performance and computational efficiency.

If a target mesh dataset is close to ideal in terms of geometry properties (does not contain holes and any other artifacts), graph interpretation of meshes could be further restricted to manifold interpretation. It is rarely the case, that real data only consists of true manifolds, however, there is a stream of works, exploring a principled extensions of CNNs to manifold data. The work of (Masci et al., 2015) introduces geodesic convolutions defined on manifolds similarly to local charting procedures. It first maps features to local patches with polar local coordinate systems and then defined convolution as matching of templates to patches for all possible angle variations in template rotations. Another work (Boscaini et al., 2016) considers the anisotropic diffusion equation on manifolds with heat kernel solutions. They allow to define local charting on manifolds in a different manner and obtain better results. Unlike graph CNN, both these methods perform convolutions in the features domain instead of the spectral domain. The work of (Monti et al., 2017) contributes a unified framework for convolution definition on non-Eucledean domains.

The approach of (Verma et al., 2018) proposes another generalization of convolutions to non-regular domains operating in the feature space. For each mesh vertex it considers a neighborhood of vertices connected to it by an edge. Then, in contrast to regular convolutions, it performs soft assignment of convolutional filters to neighbors of a given mesh vertex by means of trainable subnetwork performing softmax over a linear transformation of feature vectors corresponding to mesh vertices.

The division into separate approaches for recognition of 3D shapes using point clouds and meshes is becoming less strict at the moment. Almost every recent recognition model working

with point clouds constructs an auxiliary hierarchical structure on top of input point clouds (*e.g.* partition trees, query ball trees, trees based on nearest neighbors). These structures are used to establish hierarchical local-to-global feature calculation and aggregation schemes, which could be interpreted as graphs in feature space. At the same time, meshes are instances of graphs as well, with vertices and edges of polygons in a mesh corresponding to vertices and edges of graphs. Although, considered graphs in both cases are constructed in different domains (feature domain for point clouds, data domain for meshes), some graph-based feature extraction methods have already shown efficiency in application to both types of graphs, *e.g.* the work of (Monti et al., 2017) for graphs and manifolds and the model of (Wang et al., 2019b) for point clouds.

2.2 Generative Models

Recognition models laid the groundwork for future generative models with development of architectures suitable for processing of different 3D geometry representations and enabling effective feature extraction from them. However, feature extraction is not sufficient for generative modeling. Most of generative models based on both variational autoencoders (Kingma and Welling, 2014) and generative adversarial networks (Goodfellow et al., 2014) have two essential components: an encoder targeted at extracting low-dimensional features from target data samples, and a decoder which takes a sample from a low-dimensional space and produces a viable sample from the target data domain. In VAEs encoders are used to model approximate posterior distributions of latent variables, in GANs - to provide features for discriminators. Since encoders are conditioned on the target data, they usually exploit any structure in it for improved feature extraction, as it was discussed for 3D data recognition methods in the previous section. Decoders are aimed at solving the inverse task. They do not have any structured input and only can assume the type of the output before generation.

In encoding architectures, feature aggregation mechanisms rely the most on the inherent structural relations inside the data (max pooling for local convolutional features, feature pooling according to nearest neighbour trees for point clouds and meshes). Analogously, inversion of pooling operations is required for hierarchical decoding architectures. Thus, most of the works concentrate their effort around development of proper decoding components with effective feature unpooling mechanisms. Similarly to recognition section, we review generative models according to their choice of 3D geometry representations.

2.2.1 Generation of Occupancy Grids

Since occupancy grids rely on regular structure of data and the use of 3D CNNs, their adaptation for generative purposes is the most straightforward among 3D shape representations. The approach of (Brock et al., 2016) uses the variational autoencoding framework to model distributions of voxel occupancies. Similarly to lots of generative 2D CNNs, the decoder is implemented with transposed 3D convolutions. They allow to progressively expand the spatial resolution of output feature maps until a desired resolution is achieved. The model is optimized by minimizing the sum of cross entropy of individual per-cell Bernoulli distributions. Similar convolutional architectures for decoders are explored in the work of (Wu et al., 2016) but the resulting model is trained in an adversarial manner.

The 3D convolutional decoding architectures suitable for generation are, in fact, usually mirrored from encoding architectures with a substitution of pooling layers with unpooling

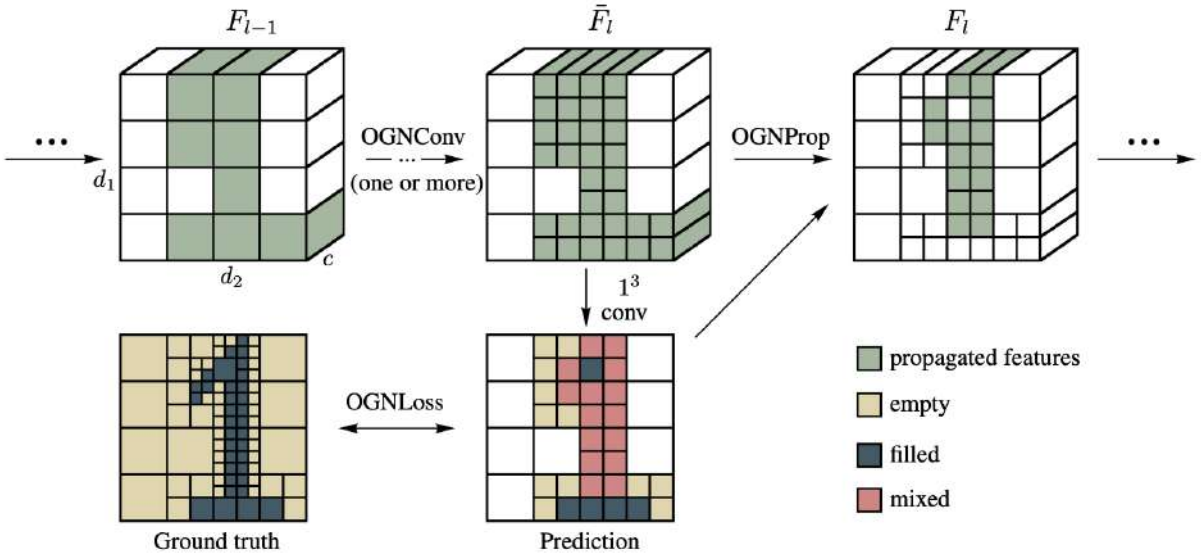

Figure 2.3: Single level of the Octree Generation Network. Spacial resolution is increased with a convolution at the locations of the cells with a mixed occupancy status. Then, a classifier decides from the new voxel which cells have an empty, filled, or mixed status. Picture is taken from (Tatarchenko et al., 2017).

analogues. That means that they suffer from the same drawbacks, namely, the limited scaling capabilities with respect to the target voxel grid resolution. To deal with this, the model of (Tatarchenko et al., 2017) proposes a 3D convolutional decoder predicting the octree corresponding to the target voxel grid instead of the explicit per-voxel occupancies. At every level of an octree each cell is classified to be empty, filled, or mixed, see Figure 2.3. Then, at the next level the same classification is performed only for the mixed cells to exploit the sparsity of the occupancy grid data.

2.2.2 Generation of Point Clouds

Generation of unordered sets is a much less explored topic compared to generation of regular grids. The work of (Achlioptas et al., 2018) starts research in this topic by application of GANs to the task. It experiments with two options: 1) direct sampling of point clouds from a GAN, 2) learning an autoencoder with similarity metric optimization first, and learning a GAN capable of sampling in the latent space of this autoencoder. A network with fully-connected layers is used to reconstruct point clouds of fixed cardinality by a final layer predicting 2048×3 features that are restructured into 2048 3D points. By construction, that limits the approach to generation of fixed-size point clouds.

The point cloud GAN model of (Li et al., 2018a) argues that simple application of Deep Sets architectures (Zaheer et al., 2017) in the decoding part of generative models is not sufficient for effective generation. Instead, the authors propose a hierarchical GAN with global per-shape and local per-point sampling procedures and suitable optimization objectives. Unlike the model of (Achlioptas et al., 2018), the decoding part of that approach uses a PointNet-like architecture conditioned on both samples from global and local noise for decoder. This allows to sample point clouds of arbitrary size.

The same underlying probabilistic model for hierarchical point cloud generation is explored in

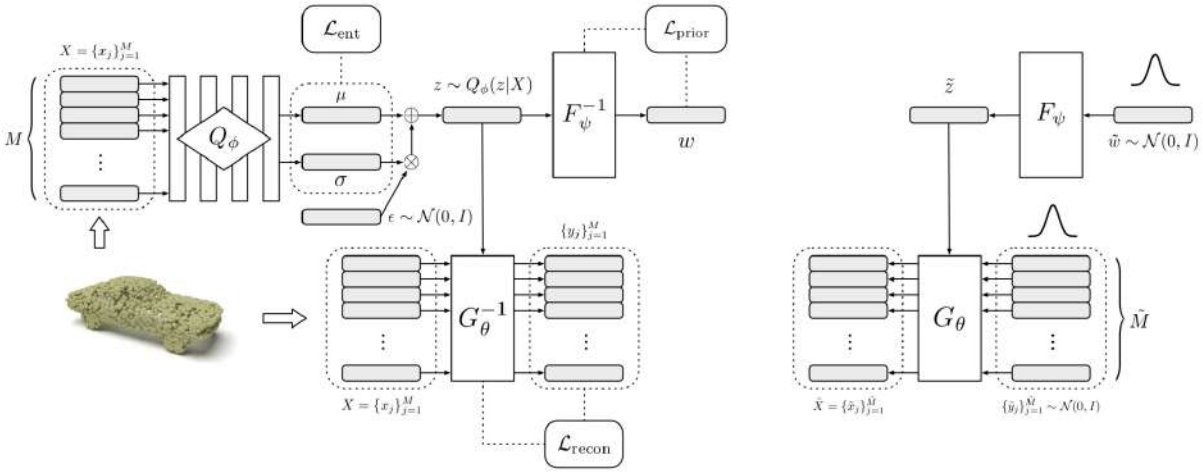

Figure 2.4: Schematic overview of the PointFlow generative model for point clouds. During training (left image): 1) point cloud encoder provides the posterior $Q_\phi(z|X)$ for the latent variable z, 2) a latent variable sample from the posterior is used to condition the continuous normalizing flow G_θ that is traversed in the reverse direction to obtain base flow distribution samples, 3) latent variable samples are also fed to a latent variable prior, implemented by an additional flow F_ψ. During sampling (right image), a sample from latent variable prior is taken and used to condition the point flow transforming samples from the base flow distribution into target surface point samples. Picture is taken from (Yang et al., 2019).

the work of (Yang et al., 2019). In this model, a global latent variable conditions a continuous normalizing flow (Chen et al., 2018) that is used to capture point cloud densities in the decoder. This flow, takes global sample and samples from the base flow distribution and transforms them to sample points from 3D shape surfaces. The networks involved in the flow use PointNet-like layers, which allows the architecture to be able to model point cloud of various sizes. Unlike PCGAN of (Li et al., 2018a), this approach explicitly implements the probabilistic model for point cloud generation, so it is trained by maximization of the variational likelihood approximation. The computational flow of the model during training and sampling can be found in Figure 2.4.

The recent work of (Cai et al., 2020) reimagines the probabilistic approach for the task. It argues that modeling exact surface distributions in 3D space is problematic, relaxes it by Gaussian smoothing to avoid discontinuity and prevent the distribution from having zero support over the 3D space containing the surface. Then, instead of learning this distribution explicitly, the model is trained to predict the gradient of the log density for given continuous locations. After training, the model can be used in an iterative procedure that moves input points to high probability locations in 3D space, corresponding to a particular 3D shape surface.

Another recent model, proposed by (Zhou et al., 2021), uses a novel probabilistic model for point cloud generation based on inversion of a diffusion process. The generation is presented as a denoising procedure with multiple intermediate latent variables conditioning each other in a sequential manner. The model is designed in a way that every posterior and prior distributions for intermediate diffusion and its inversion states have explicit Gaussian form. The complete model is trained with approximate likelihood maximization using variational lower bound for every intermediate variable. The DNN implementation of the model is based on the point-voxel convolutional layers, reviewed earlier.

2.2.3 Generation of Meshes

Early approaches for mesh generation avoided explicit mesh modeling. Instead, the mesh autoencoder of (Tan et al., 2018) models the space of deformations of densely corresponding meshes (human shapes in case of this work). A special compact representation for these deformations (Gao et al., 2016) is captured instead of full shape geometries. The fully-connected architecture of resulting autoencoder is trained by optimization of squared distances between predicted and ground truth features and a KL-divergence term between prior and posterior for the latent variable.

The deformation approach was further explored in the work of (Gao et al., 2019). The model works with meshes that contain part annotations and develops a hierarchical part-to-shape generation process. During training, each part is encoded by several parameters, indicating the existence, relation to other parts, position, symmetry and extracted features encoding explicit deformations of a template mesh. All the part descriptors are concatenated and autoencoded by a VAE to enable generation. Additional information encoded in the part feature vectors helps to aggregate and further improve generation results.

The probabilistic model of (Nash et al., 2020) is one of the first works, exploring explicit mesh modeling. It views mesh generation as joint generation of vertices and faces, and factorizes the according model into separate unconditional autoregressive generation of vertices and autoregressive generation of faces conditioned on vertices. Coordinates of vertices are also modeled autoregressively with a NN architecture based on transformers (Vaswani et al., 2017), powerful attention-based NN layers. Face generation is performed by autoregressive pointer networks (Vinyals et al., 2015). At each step the autoregressive network produces a pointer which is compared to embeddings of the input set of vertices to produce per-vertex probabilities defining the next chosen vertex in a face. Similarly to text generation autoregressive models, it also compares the pointer to tokens indicating the end of current face and the end of the overall shape mesh. The resulting model is able to produces realistic meshes but requires significant computational resources.

2.2.4 Generation of Implicit Functions

Occupancy grid representation turned out to be problematic in terms of scaling to higher grid resolutions. Point clouds, on the other hand, do not provide explicit geometry, often desired from generative and reconstruction approaches. Implicit function representations were proposed to overcome those difficulties. Two works (Chen and Zhang, 2019; Mescheder et al., 2019) simultaneously proposed to use PointNet-like MLPs to predict occupancy for continuous input 3D locations. To make occupancy predictions shape dependent, both models concatenate latent variable samples to the coordinates of query points prior to processing it with the MLPs. To enable generation, the first approach uses the adversarial framework, while the second one uses VAEs. Isosurface extraction is the one drawback of the implicit function representation that requires special attention. The work of (Chen and Zhang, 2019) proposes a naive solution: the occupancy is sampled densely over a regular grid and the Marching Cubes algorithm is applied to extract a meshed representation of the isosurface. Mescheder et al. (2019) propose an improved algorithm, based on coarse-to-fine occupancy sampling and simultaneous construction of the octree over the filled parts of the whole grid volume.

Concurrent work of (Park et al., 2019) proposes to infer signed distance function values with the same PointNet-like MLPs. Similarly to previously mentioned models, DeepSDF takes continuous point coordinates concatenated to a global latent variable sample, but predicts

the signed distance to the shape surface. Marching cubes is used to extract zero level sets explicitly representing the output geometry.

2.3 Single-View Reconstruction Models

Multiple inference applications were explored for 3D shapes including normal prediction (Guerrero et al., 2018), surface reconstruction (Erler et al., 2020), shape completion (Han et al., 2017), *etc*. However, since this thesis contributes particularly to the single-view 3D shape reconstruction (SVR) task, in this section we focus our review around selected contributions to this task.

Initial works use 3D convolutional networks to predict occupancy grids. TL-Networks (Girdhar et al., 2016) proposes a two-step training process for the task: 1) a 3D convolutional deterministic autoencoder is trained to extract low-dimensional shape space, 2) a 2D CNN image encoder is trained to infer samples from the shape space of the autoencoder. Concurrent work of (Choy et al., 2016) does not use a 3D CNN encoding component and directly trains a similar hourglass shaped architecture with 2D CNN image encoder and 3D CNN shape decoder to output occupancy grids. Additionally, the authors consider a multi-view case and propose a variation of LSTMs to fuse features extracted from different view images. Octree generation networks (Tatarchenko et al., 2017), discussed in the previous section, also propose a modification for the SVR task using 2D CNN image encoder and their OGN occupancy decoder.

The method of (Richter and Roth, 2018) proposes another compact representation for occupancy grids, that overcomes the scaling issues associated with increasing grid resolutions. Each grid is represented by shape layers: six depth maps corresponding to parallel projections to both sides along each axis. One shape layer is not sufficient to capture occluded parts of a shape, instead, several collections of depth maps are predicted and combined by unions and subtractions to capture complex geometry. Standard 2D CNNs are used in the decoders of the model to capture the shape layers.

Similar idea was explored for 3D human shapes represented as point clouds in the work of (Gabeur et al., 2019). For every ground truth pair of image and shape it considers a combination of visible and hidden depth maps obtained by ray casting from the camera to pixel locations. The visible map corresponds to the closest intersections of each ray with a surface, while the hidden map - to the farthest intersections. A hourglass shaped DNN architecture is used to infer the depth maps from an input image. The model is trained with minimization of L_1 distance between predicted and ground truth depth maps and, an additional adversarial objective, discriminating between real and fake visible and hidden depth maps.

Given camera parameters, 3D geometry of a shape can partially explain images produced by this camera: it defines a silhouette of a shape in the image and can also explain color intensity variations, since it contains shading information as well. The approach of (Yan et al., 2016) explored a possibility to use silhouette projections obtained from occupancy grid reconstructions in a differentiable manner. The overall architecture of their DNN reminds other image-to-voxel architectures, but adds a differentiable projection layer conditioned on the given camera parameters corresponding to the input image. Such projections can be used for training even in the absence of the ground truth 3D geometry information.

This idea is further developed in the work of (Tulsiani et al., 2017), where the authors present a probabilistic framework for differentiable ray consistency used for model training. In this

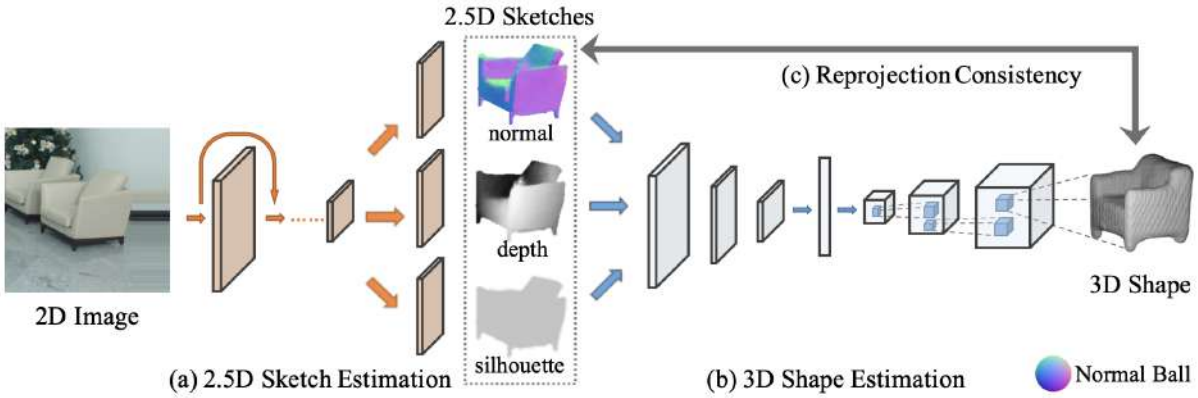

Figure 2.5: Schematic overview of MarrNet. Input images are used to extract intermediate 2.5D shape representation consisting of silhouette, depth, and normal maps. This 2.5D representation is fed to a encoder-decoder network to produce 3D occupancy grid which is used to establish consistency between the intermediate representation and final 3D shape reconstruction. Picture is taken from (Wu et al., 2017).

framework each image is interpreted as a collection of intersections of 3D shape and rays cast from the camera position. Given a grid of occupancy probabilities it is possible to calculate the probability of an intersection for every considered ray. Given these probabilities and the ground truth occupancy a cost for any incorrect intersection can be defined. Resulting loss is based on minimization of the expectation of that cost over all intersection probabilities. Similarly to the previous method, such an approach allows to reconstruct 3D shapes even without 3D geometry supervision from silhouettes and depth maps.

MarrNet (Wu et al., 2017) proposes to predict occupancy grids through an intermediate 2.5D representation. The method first learns to predict silhouettes, depth, and normal maps from input images, and then use a standard encoder-decoder architecture predicting occupancy from the intermediate representations. The resulting intermediate 2.5D representation and 3D occupancy grids are used to define consistency between depth and normal maps with occupancy grids. As the rest of the consistency based methods, this one also allows for training without ground truth 3D geometry data. Overall structure of the model is depicted in Figure 2.5.

Other works consider point cloud representation for 3D outputs of the SVR task. Point set generation networks (Su et al., 2017) construct a complex hourglass-shaped 2D CNN architecture which combines points predicted from two separate streams: 1) a fully-connected stream which outputs a fixed-size set of points, 2) a convolutional branch outputting additional set of points. The resulting model is trained by similarity metrics optimization (CD and EMD), which were, in fact, first proposed in this work prior to pure generative and autoencoding models.

Another approach is developed for point cloud reconstruction in the work of (Kurenkov et al., 2018). It proposes a two-step SVR method where, first, point clouds relevant for an input image are retrieved from the ground truth training samples; secondly, retrieved point clouds are voxelized and fed to a convolutional network together with the image to produce a regular grid of deformations. This grid is used to produce final per-point deformations for every point in the retrieved cloud. The resulting deformed clouds are compared to target shapes with similarity measures that are used for model training.

In order to improve the expressivity of the point cloud-based methods for SVR the authors

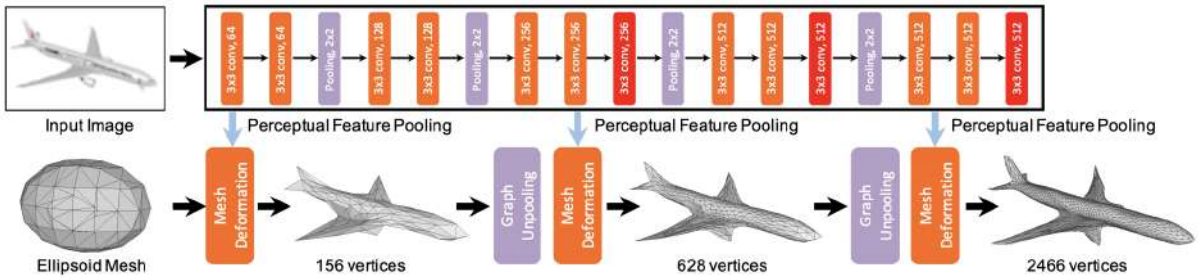

Figure 2.6: Neural network architecture of Pixel2Mesh. Image encoder extracts convolutional features from inputs and pools them locally given intrinsic camera parameters to condition mesh deformations which transform an ellipsoid mesh template into the target 3D shape. Picture is taken from (Wang et al., 2018).

of (Groueix et al., 2018) develop a patch-based prediction for point clouds. AtlasNet defines several 2D plane primitives and uses the same number of MLPs corresponding to the planes, which take samples from 2D primitives concatenated to extracted image features and produce target points in a cloud. As a result, the initial primitives are transformed into localized surface patches in 3D. The model is optimized with metric optimization as well.

The model of (Wang et al., 2018) applies the deformation framework to the output mesh representation. Initial low poly mesh ellipsoid templates are altered with deformations predicted from the input images. The deformations are interleaved with special graph unpooling operations which properly increase the complexity of the deformed mesh to obtain detailed reconstructions, see Figure 2.6. Although such an approach is effective in terms of producing smooth 3D shape mesh reconstructions, especially when trained with additional surface regularization terms, it also lacks expressivity in terms of representing various mesh topologies similarly to other methods based on template deformation.

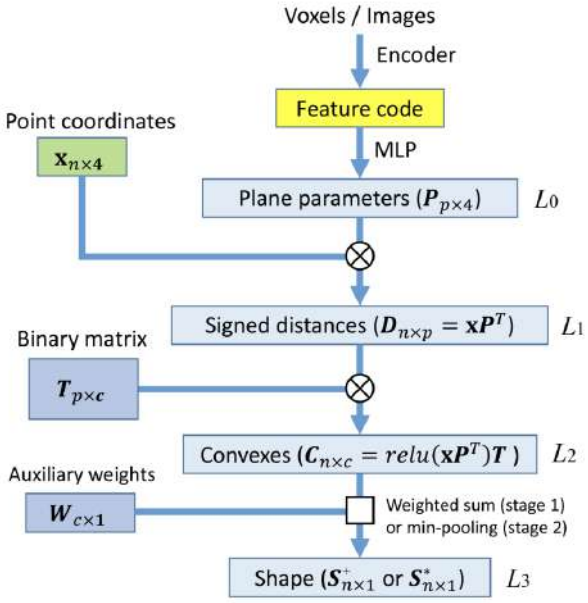

Figure 2.7: Overview of the BSP-Net. Visualization is taken from (Chen et al., 2020).

Recently, an SVR approach working with meshes was proposed in the work of (Chen et al., 2020). Similarly to implicit function methods, BSP-Net learns to predict occupancy status for input continuous locations. However, it defines the occupancy indicator function in an explainable manner using sets of predicted polygons, constructed from predicted planes. The indicator function is constructed as follows: 1) image features are used to predict 3D planes through the parameters of implicit functions defining the planes; 2) query point coordinates are fed to the plane functions to obtain signed distances; 3) signed distances are combined by a binary matrix, which at the same time encodes combinations of resulting planes forming a set of polygons; 4) the resulting

per-polygon occupancy indicators are aggregated by a min-pooling or weighted sum operation into the final shape occupancy indicator function. After training, the learned planes and connectivity can be extracted and used as an explicit mesh representation of the target shape. A schematic overview of the model can be found in Figure 2.7.

In this chapter we reviewed a number of selected recognition, generation and single-view 3D shape inference methods working with various 3D shape representations. The diversity of architecture structures and training methods of the single-view 3D shape inference methods encouraged us to propose a principled probabilistic model for the task in Chapter 3. It enabled us to analyse and compare existing design choices for SVR models and to propose our effective variant for occupancy grid representation. Our work on the probabilistic generative model for point clouds, presented in Chapter 4 was inspired by the lack of approaches working with arbitrary-size point clouds. It started prior to the release of PointFlow, the closest concurrent work of (Yang et al., 2019), so over the course of the project we decided to refocus our contributions at direct comparison and improvements over PointFlow.

CHAPTER 3

Probabilistic Reconstruction Networks for 3D Shape Inference

Various end-to-end learning strategies for 3D shape inference from images have been investigated that explore different shape representations and suitable learning architectures. In this chapter, we focus instead on the underlying probabilistic mechanisms involved and contribute a more principled probabilistic inference-based reconstruction framework, which we coin Probabilistic Reconstruction Networks. This framework expresses image conditioned 3D shape inference through a family of latent variable models, and naturally decouples the choice of shape representations from the inference itself. Moreover, it suggests different options for the image conditioning and allows training in two regimes, using either Monte Carlo or variational approximation of the marginal likelihood. Using our Probabilistic Reconstruction Networks we obtained single image 3D reconstruction results that set a new state of the art on the ShapeNet dataset in terms of the intersection over union and earth mover's distance evaluation metrics at the time of the publication. Interestingly, we obtained these results using a basic voxel grid representation, improving over recent work based on finer point cloud or mesh based representations.

This chapter is based on the following publication:

Klokov, R., Verbeek, J., and Boyer, E. (2019). Probabilistic reconstruction networks for 3D shape inference from a single image. In *BMVC*,

that was distinguished by oral presentation and received "Best Science Paper Honourable Mention" award at the conference.

3.1 Introduction

The overwhelming success of convolutional neural networks on image data (LeCun et al., 1998; Krizhevsky et al., 2012) instigated the exploration of CNNs for other problems, in particular in 3D visual computing. 3D CNNs for shapes represented with uniform voxel grids have been investigated for recognition (Wu et al., 2015; Maturana and Scherer, 2015) and generative modelling tasks (Brock et al., 2016; Wu et al., 2016). For 3D shape inference, initial works (Girdhar et al., 2016; Choy et al., 2016) successfully demonstrated the ability of 3D CNNs to produce coherent voxelized shapes given single images. This task has since

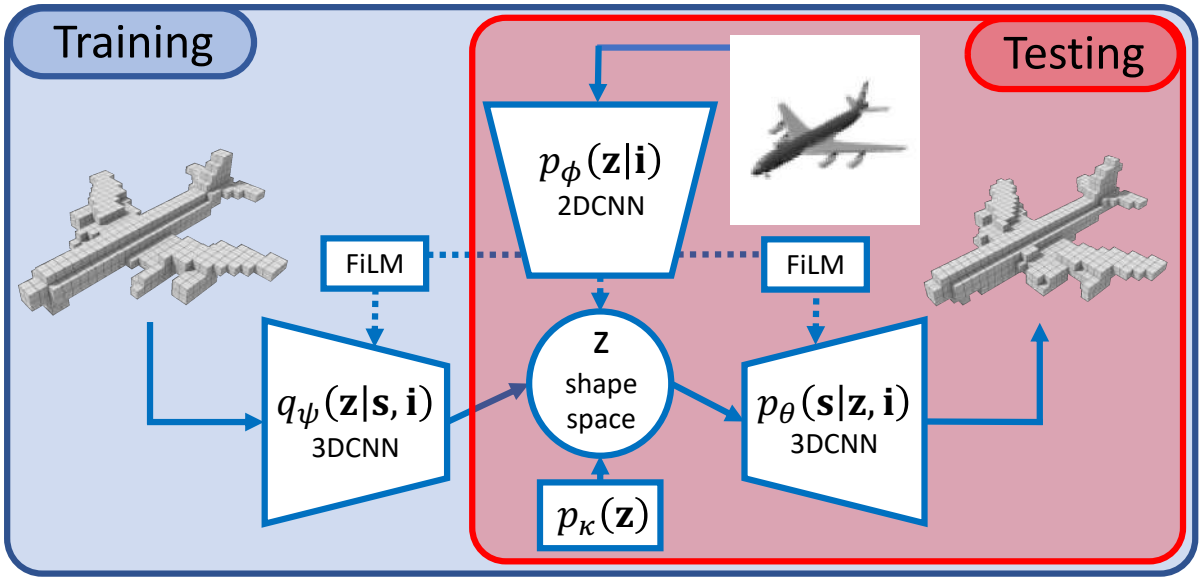

Figure 3.1: Probabilistic Reconstruction Networks for 3D shape inference from a single image. Arrows show the computational flow through the model, dotted arrows show optional image conditioning. Conditioning between 2D and 3D tensors is achieved by means of FiLM (Perez et al., 2018) layers. The inference network q_ψ is only used during training for variational inference.

gained a significant attention, as a result of its vast application field and despite its challenging ill-posed nature.

Further exploring CNNs in this context, recent works have investigated beyond straightforward adaptions of 2D CNNs to 3D voxel grids, notably to overcome the cubic complexity in time and memory associated with it. For instance, sparse representations of large voxel grid have been proposed to reduce complexity while allowing for finer shape details (Tatarchenko et al., 2017; Graham et al., 2018). Other, more scalable shape representations suitable for recognition and generation tasks have been investigated, including rendered images (Su et al., 2015), geometry images (Sinha et al., 2016), 2D depth maps (Soltani et al., 2017), point clouds (Qi et al., 2017a; Klokov and Lempitsky, 2017; Qi et al., 2017b), and graphs (Monti et al., 2017; Verma et al., 2018). Importantly, these representations come with specific network architectures and loss functions suited to the corresponding data structures.

The variety of approaches complicates comparisons and identification of the sources of improved performance. In particular, most works do not decouple the problems inherent to the task, and mix new shape representations, along with the associated network architectures and loss functions, with different image conditioning schemes, different probabilistic formulations of the shape prediction task, and in some cases the use of additional training data. This leads to possibly difficult or even unfair comparisons, due to the inability to confidently determine the source of improvements in new models, and emphasizes the need for a more systematic approach.

In this chapter we look at the single image 3D shape inference through the prism of a family of generic probabilistic latent variable models, which we term *Probabilistic Reconstruction Networks* (PRN), see Figure 3.1. The formalism encompassing these models naturally decouples different aspects of the problem including the shape representation, the image conditioning, and

the usage of latent shape space for the shape prediction. It also allows to categorize previous models for this task by their structural properties. Without loss of generality, we use voxel grids as shape representations and focus our attention on other aspects. We systematically analyze the impact of several design choices: (i) the dependency structure between the input image, the latent shape variable and the output variables; (ii) the effectiveness of training the model using Monte Carlo sampling or variational inference to approximate the log-likelihood; (iii) the effectiveness of a deterministic version of the model that suppresses any uncertainty associated with the latent variable; (iv) the effect of jointly learning the shape reconstruction model along with a generative shape model, which share their latent shape space.

For our experiments we use the ShapeNet dataset for the single image 3D shape reconstruction. We obtain excellent single image 3D reconstruction results with our Probabilistic Reconstruction Networks, setting new state-of-the-art results in terms of the IoU and EMD performance metrics. Interestingly, our results improve over works based on point-cloud and mesh-based shape representations.

In summary, our contributions are:

- a generic latent variable model for the single image 3D shape reconstruction;
- exploration of modeling options in a systematic and comparable manner;
- new state-of-the-art single image reconstruction results on the ShapeNet dataset.

In the rest of this chapter we first introduce our generic latent variable model in Section 3.2, which is then used to review and categorize previous work on shape inference from a single image in Section 3.3. We then present our experimental results and comparisons in Section 3.4.

3.2 Probabilistic Framework for 3D Shape Reconstruction

Below we present our generic latent variable model in Section 3.2.1, and describe the network architectures used for our experiments in Section 3.2.2. Although the proposed probabilistic model is agnostic to the underlying shape representation, we present it using a voxel grid representation in accordance to our experimental setup.

3.2.1 Latent Variable Model for Single Image 3D Shape Inference

Probabilistic model. We consider a shape $\mathbf{s}$ as a uniform voxel occupancy grid of a predefined resolution. Our task is to predict the shape $\mathbf{s}$ given an input image $\mathbf{i}$, *i.e.* to model $p(\mathbf{s}|\mathbf{i})$. While images and occupancy grids live in different spaces, both are representations of an underlying object that has a 3D shape and an appearance. Using this observation, we assume that an observed shape $\mathbf{s}$ has a latent parametrization $\mathbf{z}$ within a latent shape space of lower dimension, that captures shape variations. We then define our latent variable model for single image 3D shape reconstruction as:

$$p(\mathbf{s}|\mathbf{i}) = \int_{\mathbf{z}} p_\theta(\mathbf{s}|\mathbf{z},\mathbf{i})\; p_\phi(\mathbf{z}|\mathbf{i})\; \mathrm{d}\mathbf{z}, \tag{3.1}$$

where ϕ and θ are parameters of the model. This model consists of two modules: an image informed latent variable prior $p_\phi(\mathbf{z}|\mathbf{i})$, and a decoder $p_\theta(\mathbf{s}|\mathbf{z},\mathbf{i})$ that predicts the shape $\mathbf{s}$ given the image and the latent variables. Being a generic latent variable model, it allows us to decouple and study different aspects of the single image 3D shape reconstruction task.

Image Conditioning Options. In Equation (3.1) both latent variable prior and decoder are conditioned on the input image. If we drop the dependence on **i** from any module, we maintain dependence of the shape **s** on the image **i**, and obtain two alternative models:

$$p(\mathbf{s}|\mathbf{i}) = \int_{\mathbf{z}} p_\theta(\mathbf{s}|\mathbf{z})\ p_\phi(\mathbf{z}|\mathbf{i})\ \mathrm{d}\mathbf{z}, \tag{3.2}$$

$$p\,(\mathbf{s}|\mathbf{i}) = \int_{\mathbf{z}} p_\theta(\mathbf{s}|\mathbf{z},\mathbf{i})\ p_\phi(\mathbf{z})\ \mathrm{d}\mathbf{z}. \tag{3.3}$$

In the first case, we omit the image conditioning in the decoder $p_\theta(\mathbf{s}|\mathbf{z})$ and assume that the image conditioned prior $p_\phi(\mathbf{z}|\mathbf{i})$ is sufficient to obtain a valid reconstruction by the decoder. This dependency structure corresponds to an assumption of conditional independence of **i** and **s** given **z**. In the second case, we leave the image dependence in the decoder $p_\theta(\mathbf{s}|\mathbf{z},\mathbf{i})$ but use an unconditional prior for the latent variable $p_\phi(\mathbf{z})$. This corresponds to an assumption that **i** and **z** are a-priori independent. If we drop the image conditioning in both components, the model becomes a purely generative latent variable shape model:

$$p(\mathbf{s}) = \int p_\theta(\mathbf{s}|\mathbf{z})\ p_\kappa(\mathbf{z})\ \mathrm{d}\mathbf{z}. \tag{3.4}$$

Latent Variable Sampling during Training. Due to the non-linear dependencies in the integral in the models defined in equations (3.1)–(3.3), exact computation of the log-likelihood and its gradient is intractable. We consider two alternative approaches to overcome this difficulty. The first is to use a Monte Carlo approximation for Equation (3.1), as *e.g.* in (Gordon et al., 2019):

$$\ln p(\mathbf{s}|\mathbf{i}) \approx \ln \frac{1}{M} \sum_{m=1}^{M} p_\theta(\mathbf{s}|\mathbf{z}_m,\mathbf{i}), \qquad \mathbf{z}_m \sim p_\phi(\mathbf{z}|\mathbf{i}), \tag{3.5}$$

where we make use of the re-parametrization trick (Kingma and Welling, 2014; Rezende et al., 2014) to back-propagate the gradient of the log-likelihood w.r.t. ϕ through the sampling from $p_\phi(\mathbf{z}|\mathbf{i})$.

Alternatively, we can use the variational inference framework (Kingma and Welling, 2014; Rezende et al., 2014) to obtain a training signal based on more informed samples from the latent variable. We introduce a variational approximate posterior $q_\psi(\mathbf{z}|\mathbf{s},\mathbf{i})$, which is learned jointly with the prior and decoder by the maximization of the variational lower bound on the log-likelihood for Equation (3.1):

$$\mathcal{L}(\phi,\psi,\theta,\mathbf{s},\mathbf{i}) = \mathbb{E}_{q_\psi(\mathbf{z}|\mathbf{s},\mathbf{i})}[\log p_\theta(\mathbf{s}|\mathbf{z},\mathbf{i})] - \mathcal{D}_{\mathsf{KL}}(q_\psi(\mathbf{z}|\mathbf{s},\mathbf{i})\ ||\ p_\phi(\mathbf{z}|\mathbf{i})) \leq \ln p(\mathbf{s}|\mathbf{i}). \tag{3.6}$$

To evaluate this bound and its gradient, we sample from $q_\psi(\mathbf{z}|\mathbf{s},\mathbf{i})$, relying again on the re-parametrization trick to ensure back-propagation. Since the samples are conditioned on the shape, unlike the Monte Carlo approximation case, we expect this approach to be more sample efficient. On the one hand, image conditioning in the posterior may be omitted, since the posterior is already conditioned on the shape information. On the other hand, conditioning on the image may in principle further improve the accuracy of the approximate posterior, as it is based on more input information and more model parameters.

Deterministic Shape Model. We can obtain deterministic versions of the presented models by considering **z** as a function of ϕ and, if required, **i**. Although this simplifies the models, it also discards an important property. Typically, $p(\mathbf{s}|\mathbf{z})$ is factorized, with each voxel occupancy

being modelled as an independent Bernoulli, *e.g.* (Choy et al., 2016; Girdhar et al., 2016). For any deterministic latent shape vector it returns a fixed set of cell occupancy probabilities. In case of $\mathbf{z}$ being a function of $\mathbf{i}$, every input image produces a single possible set of occupancy probabilities. However, any single image of a 3D object does not contain enough information to deterministically define a single output shape, since it does not account for any possible variations in shape geometry, associated with self-occlusion. Thus a distribution of a latent variable that conditions the factorized data distribution can be used to induce dependencies among the voxel occupancies to reflect the structured ambiguity resulting from partially observed shapes. Or, in other words, different samples from an image-conditioned latent shape distribution can correspond to different configurations of the occluded geometry of predicted shapes.

In case of the variational training it is also possible to use a deterministic version of approximate posterior, and substitute the KL-divergence in Equation (3.6) with a suitable similarity measure. For example, the L_2-norm of the difference between the output of image encoder p_ϕ and the approximate posterior q_ψ. If the prior and approximate posterior distributions are modelled with factored Gaussians in probabilistic case, that corresponds to the distance between the means of these Gaussians.

Merging Unconditional Generation with Reconstruction. The models presented above can be trained along with a generative shape model, that may be of interest on its own, or used to regularize the conditional model. To achieve this, we consider a variational bound on the log-likelihood of an unconditional generative model:

$$\mathcal{L}(\kappa, \psi, \theta, \mathbf{s}) = \mathbb{E}_{q_\psi(\mathbf{z}|\mathbf{s})}[\log p_\theta(\mathbf{s}|\mathbf{z})] - \mathcal{D}_{\mathsf{KL}}(q_\psi(\mathbf{z}|\mathbf{s}) \mid\mid p_\kappa(\mathbf{z})) \leq \ln p(\mathbf{s}), \tag{3.7}$$

where $p(\mathbf{s}) = \int p_\theta(\mathbf{s}|\mathbf{z}) p_\kappa(\mathbf{z}) \mathrm{d}\mathbf{z}$. Looking at Equation (3.6) and Equation (3.7), we observe that if we omit image conditioning from the decoder and the approximate posterior in Equation (3.6), and share their parameters in both conditional and unconditional models, we can obtain a unified training objective:

$$\begin{aligned} \mathcal{L}(\kappa, \phi, \psi, \theta, \mathbf{s}, \mathbf{i}) = \mathbb{E}_{q_\psi(\mathbf{z}|\mathbf{s})}[\log p_\theta(\mathbf{s}|\mathbf{z})] &- \frac{1}{2}\mathcal{D}_{\mathsf{KL}}(q_\psi(\mathbf{z}|\mathbf{s}) \mid\mid p_\kappa(\mathbf{z})) \\ &- \frac{1}{2}\mathcal{D}_{\mathsf{KL}}(q_\psi(\mathbf{z}|\mathbf{s}) \mid\mid p_\phi(\mathbf{z}|\mathbf{i})), \end{aligned} \tag{3.8}$$

which is the average of the lower bounds on the marginal likelihood on the shape $\mathbf{s}$ with and without conditioning on the image. The term corresponding to the unconditional likelihood can be viewed as a regularization that encourages $\mathbf{z}$ from the entire latent space to correspond to realistic shapes, instead of just the $\mathbf{z}$ in the support of the conditional distributions $p(\mathbf{z}|\mathbf{i})$ on latent coordinates given an input image. Note that in the generative model neither the decoder $p_\theta(\mathbf{s}|\mathbf{z})$ nor the encoder $q_\psi(\mathbf{z}|\mathbf{s})$ are conditioned on the image. These can thus be shared with the single image reconstruction model if the latter is not conditioned on the image in these components.

3.2.2 Network Architectures

Although our probabilistic model is not representation specific, we focus on the voxel grid shape representation, and leave the comparison to alternative representations for the future work. Thus, we implement the different conditional distributions in our model as 2D and 3D CNNs that output the parameters of distributions on the latent variable $\mathbf{z}$, or on the voxel occupancies. In particular,

- The unconditional Gaussian latent prior $p_\kappa(\mathbf{z})$ is characterized by κ that consists of means and variances for all latent dimensions implemented as trainable parameters.
- The image conditioned prior $p_\phi(\mathbf{z}|\mathbf{i})$ is a 2D CNN, consisting of six blocks of pairs of convolutions: a standard and a strided one, interleaved with batch normalization and point-wise non-linearities, followed by two fully-connected layers. It processes input images into the means and variances of a factored Gaussian on $\mathbf{z}$.
- The shape conditioned variational posterior $q_\psi(\mathbf{z}|\mathbf{s},\mathbf{i})$ is a 3D CNN consisting of an initial 3D convolution and a series of four modified residual blocks (He et al., 2016a,b), each using an additional $1\times1\times1$ convolution instead of identity and feature map concatenation instead of summation and each followed by $2\times2\times2$ spatial average pooling. The final convolutional features are fed to two additional fully-connected layers. This encoder processes input shapes into the means and variances of a factored Gaussian on $\mathbf{z}$. Image conditioning in this and the next module is discussed further below.
- The 3D deconvolutional decoder $p_\theta(\mathbf{s}|\mathbf{z},\mathbf{i})$ is mirrored from the approximate posterior q_ψ, with the pooling being substituted by the $2\times2\times2$ upscaling by trilinear interpolations, producing the parameters of Bernoulli distributions on the voxel occupancies from a latent variable input.

Image conditioning in the two latter modules is inspired by the FiLM conditioning mechanism (Perez et al., 2018). Intermediate 2D feature maps from the first five convolutional blocks of the image encoder $p_\phi(\mathbf{z}|\mathbf{i})$ are averaged spatially, transformed by two additional fully-connected layers into weights and biases, that are used to scale the outputs of each corresponding batch-normalized collection of 3D feature maps resulting in one scaling operation per each 3D residual convolutional network block in the shape encoder, the latent variable decoder, or both. Instead of affine transformation used in FiLM, we use non-negative scaling weights by predicting them in logarithmic scale, in our experiments this resulted in more stable training and slightly better results. See Figure 3.1 for a schematic overview of the model architecture.

To ensure fair comparison between the variations of the model we use identical architectures for every component concurring in different models, except for additional fully-connected layers associated with the different image conditioning options. When we include an unconditional generative shape model and optimize Equation (3.8), we share the decoder $p_\theta(\mathbf{s}|\mathbf{z})$ and the variational posterior $q_\psi(\mathbf{z}|\mathbf{s})$ between the conditional and unconditional models. Exact architectures, training procedures and their hyperparameters are available on the implementation code page.[1]

3.3 Related Work

In this section we review related work on single image 3D shape reconstruction, and relate it to our generic latent variable model presented in the previous section.

Representations for 3D Shape Inference. The majority of works studying the inference-based single image 3D shape reconstruction introduce new shape representations and suitable neural network architectures. The seminal works (Choy et al., 2016; Girdhar et al., 2016) used 3D CNNs to predict voxel occupancy grids. To reduce the computational complexity of the voxel grid representation an architecture to process octrees computed on top of the voxel grids was proposed in (Tatarchenko et al., 2017). The authors of (Richter and Roth,

[1] https://github.com/Regenerator/prns

2018) proposed to use a set of six depth maps to represent voxel grids and to combine a series of such sets in a nested manner to model non-trivial shapes. 2D CNNs were combined with fully-connected networks to output point clouds in (Su et al., 2017). These point clouds were learned by optimizing Chamfer distance or differentiable approximation of earth mover's distance. In (Wang et al., 2018) the authors applied graph-convolutional networks (Bronstein et al., 2017) to the mesh-based shape representation. Other work (Shin et al., 2018) proposed to predict multiple depth maps and corresponding silhouettes and fuse them into meshes by post-processing with Poisson reconstruction algorithm.

Although related in their overall goals, these approaches are difficult to compare since they use target shapes approximated to different degrees. Ideally, a fair comparison across shape representations should be performed while maintaining the same level of granularity across representations, and for different levels, since it is possible that some representations work better for rough shape reconstruction, while other are best for detailed reconstruction.

Image-Shape Consistency and Additional Data. Another significant stream of works originates from the idea of ensuring consistency between input data and target shapes. Initial work (Yan et al., 2016) introduced the consistency between 3D shapes and their silhouettes produced by different viewpoints in a form of a loss function. Similar ideas were investigated in (Wiles and Zisserman, 2017). This approach was expanded in (Tulsiani et al., 2017) by the use of differentiable ray tracing in the loss function, ensuring correspondence of inferred voxelized shapes to foreground segmentation masks and depth images. In (Wu et al., 2017) the idea was developed even further and introduced a two-step reconstruction framework. The first part of the model is trained to infer 2.5D shape sketches (unions of segmentation, depth and normal maps) from images, while the second is separately trained to predict shapes from 2.5D sketches. Both components are then fine-tuned, using reprojection consistency.

A probabilistic framework for image generation conditioned on a latent shape variable and an additional latent variable for the shape pose was proposed in (Henderson and Ferrari, 2018). This framework was used to train an underlying 3D mesh generator with the help of differentiable rendering of 3D meshes into images. Differentiable point clouds (Insafutdinov and Dosovitskiy, 2018) closed the consistency loop between inferred point clouds and input images, by rendering point clouds as images and minimizing loss between such renderings and input images.

Similarly to the previous class of models, these methods also enrich available training data by considering different forms of additional data: camera information, 2.5D sketches, *etc*. This, again, makes comparison problematic, since it is not always clear what the impact of the additional training data is.

Relations to Our Framework. In addition to the work discussed above, given similarities between VAEs and GANs (Hu et al., 2018), our work is also related to adversarial approaches involving shape discriminators (Wu et al., 2016; Smith and Meger, 2017; Wu et al., 2018). In Table 3.1 we organize related work in terms of how it fits into our generic probabilistic reconstruction framework, abstracting away from implementations of various components.

Firstly, we group existing works with respect to the different image conditioning options. Most works condition the latent variable. That usually corresponds to having a convolutional image encoder inferring a latent vector or distribution. Others directly condition the target data distribution. In that case, features from image encoders are somehow fused with the intermediate features of 3D shape decoding subparts of networks. Besides the general image conditioning structure, models differ in several other important properties: if they use

Dependencies	Sampling	Det.	Disc.	References
$p(\mathbf{s}\|\mathbf{z})p(\mathbf{z}\|\mathbf{i})$	$p(\mathbf{z}\|\mathbf{i})$	✓	✗	(Choy et al., 2016; Yan et al., 2016; Wu et al., 2017) (Tulsiani et al., 2017; Wiles and Zisserman, 2017) (Tatarchenko et al., 2017; Shin et al., 2018) (Richter and Roth, 2018; Insafutdinov and Dosovitskiy, 2018)
			✓	(Wu et al., 2018)
		✗	✗	(Henderson and Ferrari, 2018)
			✓	(Wu et al., 2016; Smith and Meger, 2017)
$p(\mathbf{s}\|\mathbf{z},\mathbf{i})p(\mathbf{z})$	$p(\mathbf{z})$	✓	✗	(Wang et al., 2018)
		✗		(Su et al., 2017)
$p(\mathbf{s}\|\mathbf{z})p(\mathbf{z}\|\mathbf{i})$	$q(\mathbf{z}\|\mathbf{s})$	✓	✗	(Girdhar et al., 2016)
		✗		(Mandikal et al., 2018)

Table 3.1: Overview of how related work fits into our probabilistic reconstruction framework. The first column indicates which components of corresponding models are conditioned by input images; the second - how the latent variable is sampled/inferred during training; the third - if a deterministic or probabilistic objective was used during training; the fourth - if adversarial training is used with the help of a discriminator.

variational inference with the approximate shape conditional posterior of the latent variable (that corresponds to using additional 3D shape encoder during training); if the variables of the model are inferred deterministically from input images or are sampled from predicted distributions (that also influences the choice of the loss functions during training); if the adversarial training approach is used either instead or in combination with other training procedures with the help of real/fake 3D shape discriminator.

We see that most previous works use a dependency structure where the latent variable is inferred from the image, and the shape decoder only depends on the latent variable and not on the image. Moreover, most works rely on deterministic models, except for GAN-based approaches of (Wu et al., 2016; Smith and Meger, 2017), the point cloud based approaches of (Su et al., 2017; Mandikal et al., 2018), and the mesh based method of (Henderson and Ferrari, 2018). Finally, only TL-Networks (Girdhar et al., 2016) and 3D-LMNet (Mandikal et al., 2018) make use of the variational inference (or its deterministic analogue) for shape modelling.

3.4 Experiments

In this section we present the experimental setup, our quantitative and qualitative evaluation results, as well as our analysis of these results.

3.4.1 Dataset, evaluation metrics, and experimental details

Dataset. We evaluate PRNs on a subset of the ShapeNet dataset (Chang et al., 2015) introduced in (Choy et al., 2016). It contains about 44k 3D shapes from 13 major categories of ShapeNet dataset represented as voxel grids of resolution 32^3, as well as renderings from 24

Dependencies	Sampling	Deterministic	Shape model	IoU↑ (0.5)	IoU↑ (0.4)
$p(\mathbf{s}\|\mathbf{z})p(\mathbf{z}\|\mathbf{i})$	$p(\mathbf{z}\|\mathbf{i})$			63.7	65.0
$p(\mathbf{s}\|\mathbf{z},\mathbf{i})p(\mathbf{z}\|\mathbf{i})$	$p(\mathbf{z}\|\mathbf{i})$			64.6	65.6
$p(\mathbf{s}\|\mathbf{z},\mathbf{i})p(\mathbf{z})$	$p(\mathbf{z})$			64.0	65.0
$p(\mathbf{s}\|\mathbf{z})p(\mathbf{z}\|\mathbf{i})$	$q(\mathbf{z}\|\mathbf{s})$			**65.9**	**66.2**
$p(\mathbf{s}\|\mathbf{z},\mathbf{i})p(\mathbf{z}\|\mathbf{i})$	$q(\mathbf{z}\|\mathbf{s})$			64.8	65.3
$p(\mathbf{s}\|\mathbf{z})p(\mathbf{z}\|\mathbf{i})$	$q(\mathbf{z}\|\mathbf{s},\mathbf{i})$			65.4	65.8
$p(\mathbf{s}\|\mathbf{z})p(\mathbf{z}\|\mathbf{i})$	$q(\mathbf{z}\|\mathbf{s})$	✓		63.4	63.7
$p(\mathbf{s}\|\mathbf{z})p(\mathbf{z}\|\mathbf{i})$	$q(\mathbf{z}\|\mathbf{s})$		✓	65.6	66.1

Table 3.2: Evaluation results for variations of PRN. Monte Carlo training uses samples from the unconditional or image-informed prior, while variational training relies on samples from the shape-conditioned approximate posterior. We report IoU under two occupancy probability thresholds τ.

different randomized viewpoints as 137^2 images. Following Choy *et al*., we use 80% of the shapes from each category for training and remaining shapes for testing.

Evaluation Metrics. We evaluate using the standard intersection-over-union metric defined in Equation (1.1), which averages the per-category IoU metric between the inferred shape and the ground-truth voxel representation (Choy et al., 2016). In addition, to allow comparison to recent work based on point-cloud and mesh based representations, we also report the Chamfer distance and earth mover's distance, defined in Equation (1.2) and Equation (1.3), computed using the code of (Sun et al., 2018), where we removed the square root from the distance computations in CD to make it comparable to the related work. In particular, each ground truth and predicted voxel grid is mapped to a point cloud by sampling the surface induced using the marching cubes algorithm (Lewiner et al., 2003). We then compute the CD and EMD on the resulting point clouds.

Training and Evaluating. When using either Monte Carlo approximation or a variational objective function, we always use a single sample to compute the gradients during training. A unified training protocol is used for all the models: all components are trained simultaneously (contrary to (Girdhar et al., 2016; Wu et al., 2018; Mandikal et al., 2018)), with the AMSGrad (Reddi et al., 2018) optimizer with decoupled weight decay regularization (Loshchilov and Hutter, 2019) with step-like scheduling for learning rate and weight decay parameter, and restarts of gradient moments accumulation at the beginning of each step.

During testing, we use a deterministic approach. In particular, we take the means of the conditional distributions rather than samples from them. We found this to significantly improve the reconstruction quality, compared to sampling.

3.4.2 Experimental results

Evaluation of PRN Variants. To explore the various possibilities of our general latent variable model, we consider three options to condition on the image: (i) using the latent space to carry all image information: $p(\mathbf{s}|\mathbf{z})p(\mathbf{z}|\mathbf{i})$, (ii) using additional conditioning of the decoder on the image: $p(\mathbf{s}|\mathbf{z},\mathbf{i})p(\mathbf{z}|\mathbf{i})$, and (iii) using an uninformed prior on the latent variable:

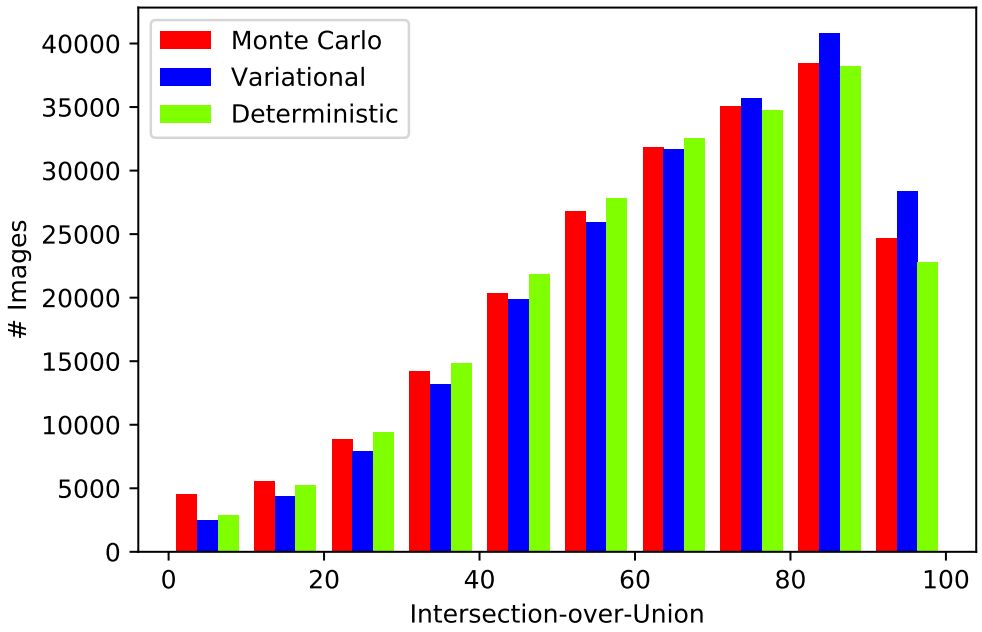

Figure 3.2: Histogram of IoU values on the ShapeNetAll test set for the Monte Carlo, variational, and deterministic models. See text for details. Bins of size 10 from 0 to 10, then 10 to 20, *etc*.

$p(\mathbf{s}|\mathbf{z}, \mathbf{i})p(\mathbf{z})$. To train the models we either approximate the integral in Equation (3.1) directly with Monte Carlo samples from the prior on $\mathbf{z}$, or with a variational lower bound and samples from the variational posterior. We also test a deterministic model, where the distribution on the latent variable is replaced by a deterministic function. Finally, we consider the option to train the model jointly with an an unconditional generative model. In Table 3.2 we present the results, using two thresholds τ on the voxel occupancy probability: the neutral 0.5, and following (Choy et al., 2016) the looser 0.4 which overall leads to improved IoU scores.

In case of the Monte Carlo approximation (top three rows), additional image conditioning in the decoder improves the results. Conditioning both the latent variable prior and the decoder on the image achieves best results, suggesting that these different pathways to use the image are complementary.

The use of variational training consistently improves the results over the Monte Carlo approximation. In this case, the additional image conditioning of the decoder or the variational posterior, see line five and six, is not effective and even somewhat reduces the performance. This is contrary to the results obtained with Monte Carlo sampling; in the latter case the sampling inefficiency is probably partially compensated by the additional conditioning pathway. Variational inference leads to more accurate samples, which obviates the need for the additional image conditioning (at least for the chosen mechanism of the additional image conditioning).

We also consider a deterministic variant of our best performing model, which resembles the TL-network of (Girdhar et al., 2016). The results show that probabilistic handling of the latent variable reduces overfitting in the model and leads to IoU of 2.5 points higher. Finally, we also tested the training with a joint generative shape model, which TL-Networks used as pre-training. Although we did not observe a significant effect due to the joint training with a

generative shape model, it does offer additional functionality by being able to sample shapes, or compute their likelihoods under the model.

In Figure 3.2 we provide a histogram of the IoU scores obtained across the shapes in the ShapeNet test set using three of the model evaluated in Table 3.2:

- Using $p(\mathbf{v}|\mathbf{z})p(\mathbf{z}|\mathbf{i})$, with Monte Carlo training (Table 3.2, row 1).
- Using $p(\mathbf{v}|\mathbf{z})p(\mathbf{z}|\mathbf{i})$, with variational training (Table 3.2, row 4).
- Using $p(\mathbf{v}|\mathbf{z})p(\mathbf{z}|\mathbf{i})$, with deterministic training (Table 3.2, row 7).

For each shape in the test set there are 24 views, giving a total of about 210k shape inferences.

The histograms show that the variational learning approach leads to more accurate reconstructions, leading to the largest number of reconstructed shapes in the last three bins for shape with $> 70\%$ IoU. For all other bins of less accurate results, the variational method has the smallest number of shapes. Compared to the deterministic model, Monte Carlo training leads to more accurate reconstructions, but also to more very poor reconstructions.

Comparison to the State of the Art. In Table 3.3 we compare to earlier state-of-the-art approaches. All methods use the same input images, but use slightly different image preprocessing: 3D-R2N2 uses random cropping, 3D-LMNet central cropping, AtlasNet crops and resizes, while PSGN and Pixel2Mesh resize the image. As OGN, we use original images, but also add a grey-scale version of each image as a fourth input channel.

In Table 3.3 we report Chamfer distance and earth mover's distance computed for two numbers of both predicted and ground truth points and two different data normalization scales. We do it for the ease of comparison with different groups of related works. We, firstly, compute the metrics for 1024 ground truth and predicted points and rescale point clouds so that the original 3D shapes fit into unit cubes, which corresponds to the evaluation setup of (Mandikal et al., 2018). Secondly, we evaluate for 2500 ground truth and predicted points and rescale clouds so that the original 3D shapes fit into unit radius spheres to properly compare to AtlasNet and Pixel2Mesh. The second scale is approximately two times larger compared to the first

Model	Image res.	Output	IoU↑	CD↓	EMD↓
3D-R2N2 (Choy et al., 2016)	127^2	voxels 32^3	56.0	—	—
OGN (Tatarchenko et al., 2017)	137^2	voxel octrees 32^3	59.6	—	—
PSGN (Su et al., 2017)	128^2	points 1024	64.0	5.62	7.75
3D-LMNet (Mandikal et al., 2018)	128^2	points 1024	—	5.40	7.00
PRN*	137^2	voxels 32^3	**66.2**	**4.42**	**6.32**
AtlasNet (Groueix et al., 2018)	224^2	points 2500	—	**53.4**	12.54
Pixel2Mesh (Wang et al., 2018)	224^2	meshes 2466	—	59.1	13.80
PRN†	137^2	voxels 32^3	**66.2**	75.6	**11.00**

Table 3.3: Comparison of PRN to the state-of-the-art. CD and EMD are multiplied by 100. For proper comparison PRN results are presented for two different data normalization: * - point clouds obtained from ground truth and reconstructed voxel grids are scaled to fit into unit cubes, † - point clouds are scaled to fit into unit radius spheres.

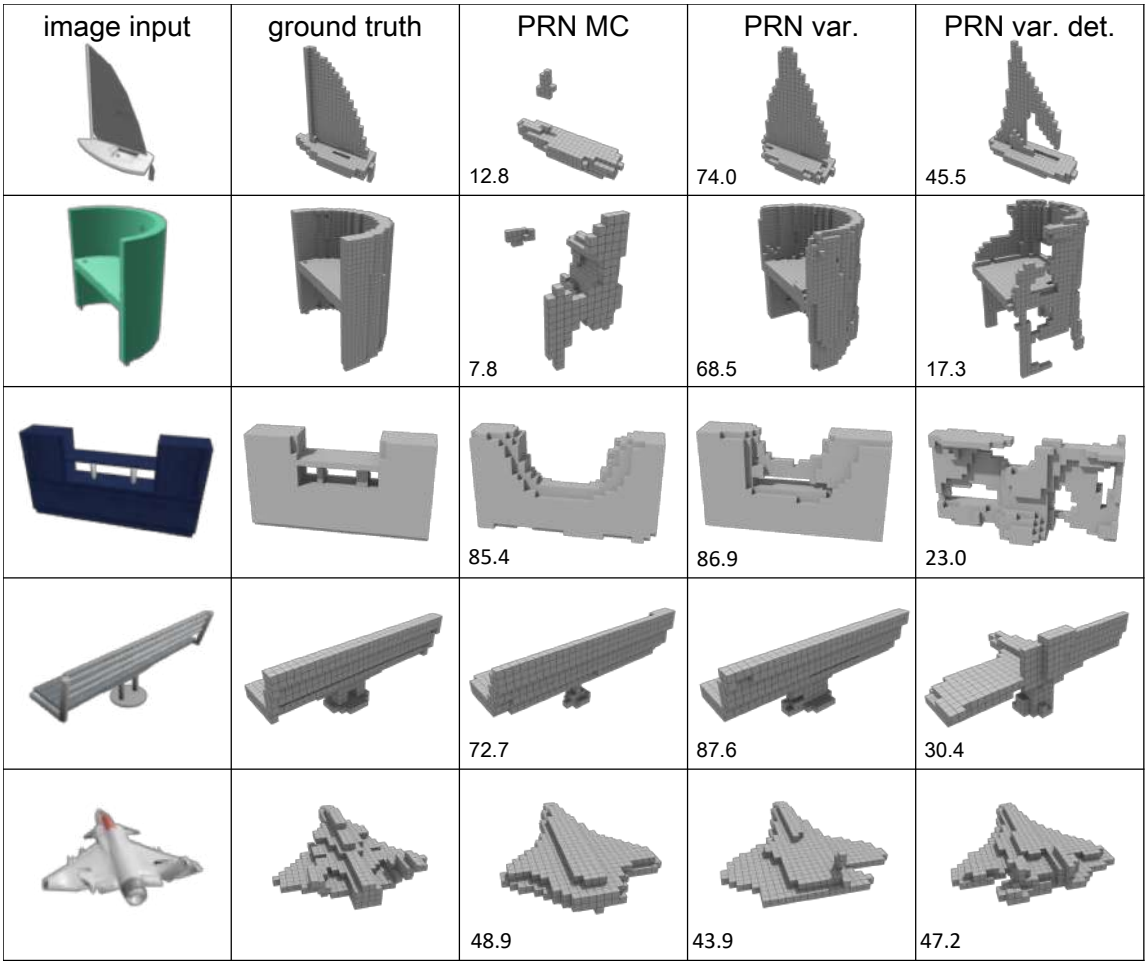

Figure 3.3: Qualitative reconstruction results for three variants of PRNs. The IoU metric is stated near each reconstruction.

one, which translates into approximately two times larger distances between points d. Given that CD is proportional to $2d^2$ and EMD to d, such a difference in the second scale should increase the metric values approximately eight and two times accordingly.

It is important to underline that both CD and EMD depend on the number of predicted and ground truth points used for evaluation. Larger numbers of points decrease the distances between closest points and result into lower metric values. For the reference, we recomputed CD using 1024 predicted and 16384 ground truth points and obtained a better value: 3.90. That underlines the need for unified evaluation protocol among different models when evaluating for these metrics.

Our PRN obtains excellent results, and significantly improves over previous state-of-the-art results in terms of IoU and EMD, including methods based on point cloud and mesh representations. Point-based approaches, such as AtlasNet, use loss functions based on the Chamfer distance, and so naturally perform well in terms of this metric, but this does not per se transfer to better performance in the other metrics. In our case, we do not explicitly optimize for either of these metrics, relying on the binary cross-entropy for the voxel occupancies instead, and yet obtain best results in two of the three metrics.

Qualitative Reconstruction Results. In Figure 3.3 we provide a selection of qualitative reconstruction results. We show results for the models in rows one, four and seven in Table 3.2,

i.e. with the $p(\mathbf{s}|\mathbf{z})p(\mathbf{z}|\mathbf{i})$ dependency structure, using Monte Carlo (PRN MC) and variational (PRN var.) training, and the deterministic version of the latter (PRN var. det.). We show four examples where the variationally trained model is the best, and one case where it is the worst. Overall, the variational model output fewer failed reconstructions, as well as more detailed reconstructions, compared to more failures and over simplified reconstructions from the model trained with Monte Carlo. For reference, the average IoU score of the variationally trained model is 66.2 (median 69.9), which corresponds to a fairly accurate reconstruction level, in particular given the challenging nature of the task. We also provide randomly sampled examples of reconstructions in Figure 3.4.

3.5 Conclusion

In this chapter we presented Probabilistic Reconstruction Networks, a generic probabilistic framework for 3D shape inference from single image. This framework naturally decouples different aspects of the problem, including the shape representation, the image conditioning structure, and the use of the latent shape space. In our experiments with voxel-grid representations, we systematically explored the impact of image conditioning, Monte Carlo *vs*. variational likelihood approximation for training, the stochastic nature of the latent variable, and joint training with a generative shape model. We obtained single image shape reconstruction results that surpass the previous state of the art in terms of the IoU and EMD performance metrics, and outperform recent work based on point-cloud and mesh-based representations.

Given the interpretation of the inference-based reconstruction as an instance of conditional generation, future work might include further adaptation of the generative modelling approaches to the task, as well as the investigation of different shape representations within the proposed framework.

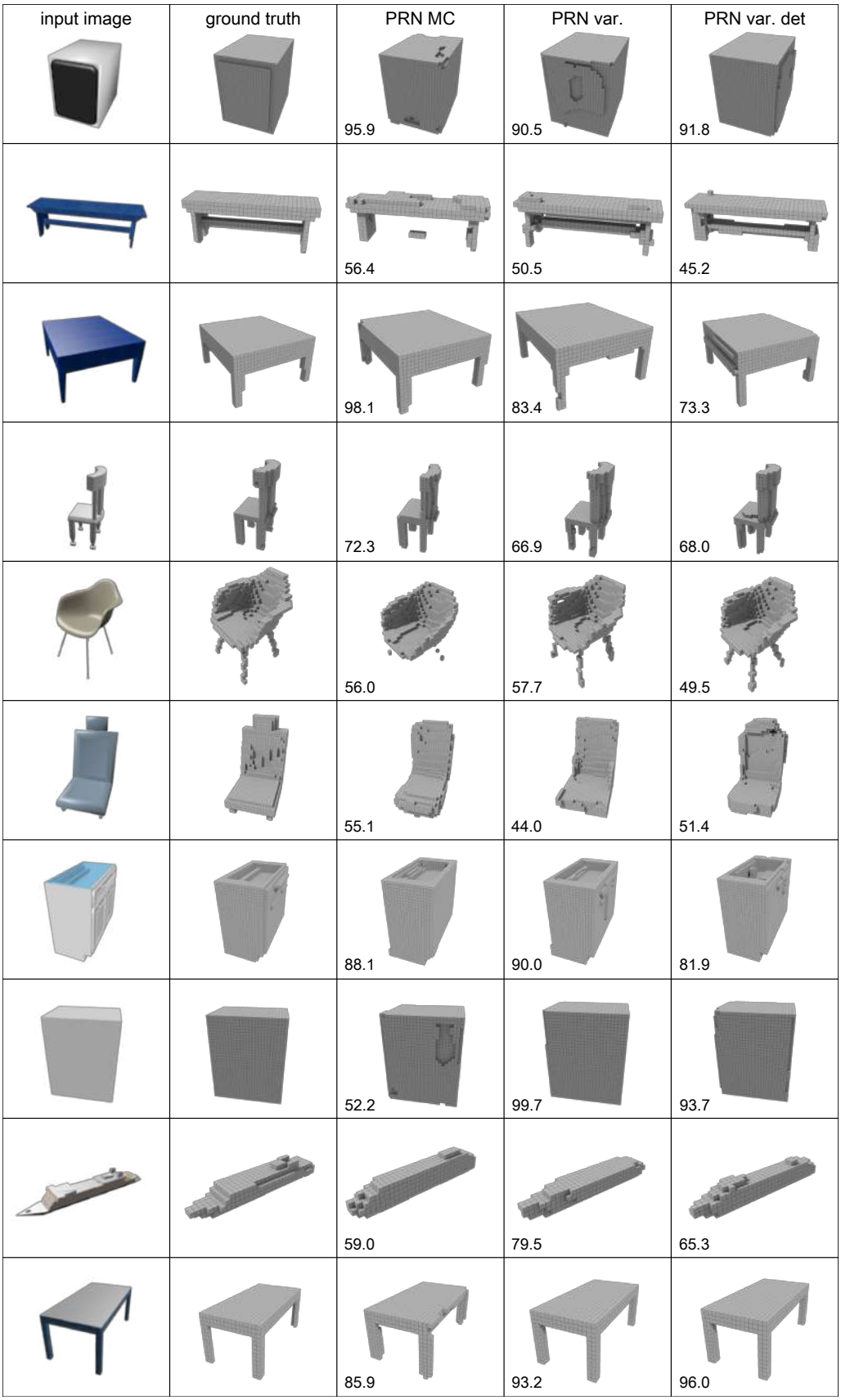

Figure 3.4: Randomly sampled reconstructions with three variants of PRNs. The IoU metric is stated near each reconstruction.

CHAPTER 4

Discrete Point Flow Networks for Efficient Point Cloud Generation

Generative models have proven effective at modeling 3D shapes and their statistical variations. In this chapter we investigate their application to point clouds, a 3D shape representation widely used in computer vision for which, however, only few generative models have yet been proposed. We introduce a latent variable model that builds on normalizing flows with affine coupling layers to generate 3D point clouds of an arbitrary size given a latent shape representation. To evaluate its benefits for shape modeling we apply this model for generation, autoencoding, and single-view shape reconstruction tasks. We improve over recent GAN-based models in terms of most metrics that assess generation and autoencoding. Compared to recent work based on continuous flows, our model offers a significant speedup in both training and inference times for similar or better performance. For single-view shape reconstruction we also obtain results on par with state-of-the-art voxel, point cloud, and mesh-based methods.

This chapter is based on the following publication:

Klokov, R., Boyer, E., and Verbeek, J. (2020). Discrete point flow networks for efficient point cloud generation. In *ECCV*.

4.1 Introduction

Generative shape models are used in numerous computer vision applications where they allow to encode 3D shape variations with respect to different attributes, such as shape classes or shape deformations, as well as to infer shapes from partial observations, for instance from a single or a few images. Central to shape models is the representation chosen for shapes that can be extrinsic, for example the ubiquitous voxels and octrees, or intrinsic as with meshes and point clouds. While extrinsic representations enable relatively straightforward extensions of 2D deep learning techniques to 3D, they suffer from their inherent trade-off between precision and complexity. This is why 3D shapes are often represented using intrinsic models, among which point clouds are a natural and versatile solution, serving as a basis for many 3D capturing methods, including most multi-view stereo and range sensing methods, *e.g.* kinect (Zhang, 2012).

Following the success of CNNs for 2D computer vision problems, many deep learning models have been proposed that can handle 3D data. This includes works on voxel grids (Maturana

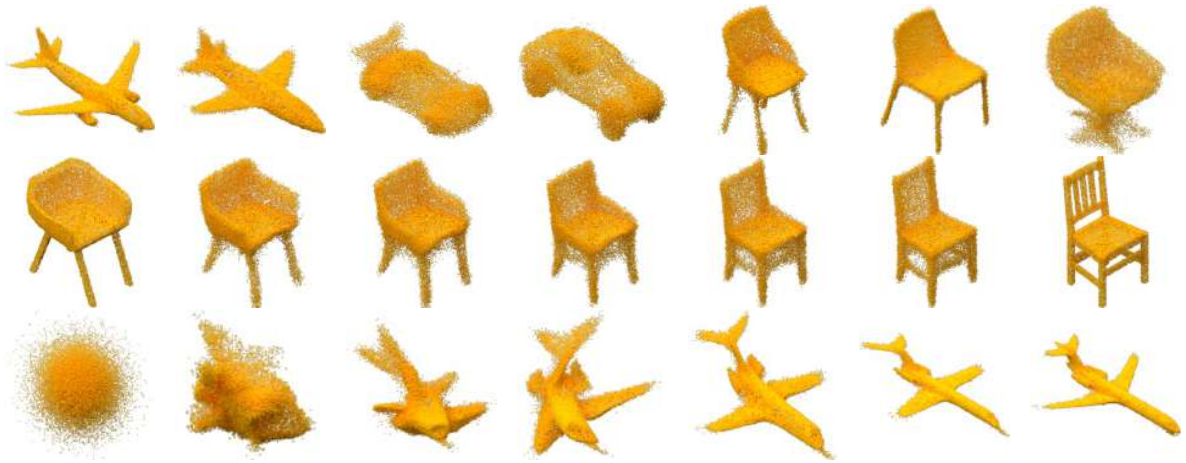

Figure 4.1: Top: Point clouds sampled from DPF-Net for the ShapeNet classes *airplane*, *car*, and *chair*. Middle: Latent space interpolation between two point clouds from the test set. Bottom: Deformation of points across the flow steps.

and Scherer, 2015; Wu et al., 2015; Choy et al., 2016; Girdhar et al., 2016; Brock et al., 2016; Wu et al., 2016; Graham et al., 2018; Klokov et al., 2019), octrees (Riegler et al., 2017; Tatarchenko et al., 2017), meshes (Monti et al., 2017; Verma et al., 2018; Wang et al., 2018; Henderson and Ferrari, 2018), point clouds (Qi et al., 2017a; Fan et al., 2017; Klokov and Lempitsky, 2017; Qi et al., 2017b; Insafutdinov and Dosovitskiy, 2018; Mandikal et al., 2018; Wang et al., 2019a), and implicit functions (Michalkiewicz et al., 2019; Park et al., 2019). While they provide effective tools to build predictive models of 3D shapes, *e.g*. from a single image, we investigate in this paper the less extensively explored and more generic problem of probabilistic generative 3D shape modeling.

Significant advances have been made in generative modeling of natural images using deep neural networks with convolutional architectures. These models have been extensively studied to model natural images, using convolutional network architectures. Consequently, they can easily be adapted to generative shape models which are based on extrinsic representations, using regular 3D convolutional layers (Wu et al., 2016; Brock et al., 2016; Klokov et al., 2019). On the other hand, their extensions to intrinsic representations, such as point clouds and meshes, are less obvious and, to the best of our knowledge, so far, only Yang *et al*. (Yang et al., 2019) have studied generative models from which arbitrary size point clouds can be sampled without any conditioning information.

We explore a hierarchical latent variable model that treats the points as exchangeable variables, which allows us to model and sample point clouds of an arbitrary size. Within this framework, each point cloud is considered as a sample from a shape-specific distribution over the 3D surface of the object, and these distributions are embedded in a latent space. To sample a point cloud, first, a vector is sampled in the latent shape space, and then, any desired number of 3D points can be sampled i.i.d. conditioned on the latent shape representation. Our model shares the high-level structure with PointFlow (Yang et al., 2019), but differs in the underlying network architectures, reducing the training and sampling time by more than an order of magnitude. In particular, our model builds on discrete normalizing flows with affine coupling layers (Dinh et al., 2017) rather than continuous flows, and FiLM conditioning layers (Perez et al., 2018) to construct a flexible density on 3D points given the latent shape representation. In Figure 4.1 we illustrate diverse point clouds sampled from our class-specific models, interpolation between point clouds in the learned latent shape space, and the sequential

process by which the discrete normalizing flow warps the points to obtain the final shape.

We evaluate generative and autoencoding capabilities of our model, as well as its use for single-view shape reconstruction. We obtain similar or better performance compared to GAN-based models in terms of metrics that assess generation and autoencoding. Compared to recent work based on continuous flows, our model offers a significant speedup, for similar or better performance. For single view reconstruction our model performs on par with state-of-the-art methods, yet allows to reconstruct with arbitrarily large point clouds. Moreover, we analyze various design choices regarding data splitting and normalization.

4.2 Related work

Generative Models. Deep neural networks have sparked significant progress in generative modeling. The most widely adopted models are variational autoencoders (VAEs) (Kingma and Welling, 2014; Rezende et al., 2014), generative adversarial networks (GANs) (Goodfellow et al., 2014; Karras et al., 2018), and normalizing flows (Deco and Brauer, 1995; Dinh et al., 2017). All three approaches share the basic principle of defining a latent variable $\mathbf{z}$ with a simple prior, *e.g.* unit Gaussian, and construct a complex conditional $p(\mathbf{x}|\mathbf{z})$ on data $\mathbf{x}$ by means of deep neural networks. Maximum likelihood training of the resulting marginal $p(\mathbf{x})$ is generally intractable due to the non-linearities. To train the model, VAEs rely on an amortized inference network that produces a variational posterior $q(\mathbf{z}|\mathbf{x})$. GANs, on the other hand, use a discriminator network to distinguish training examples and model samples, and use it as a signal to train the generative model. Alternatively to previous approaches, normalizing flow models rely on invertible neural network architectures to avoid the intractability of the marginalization altogether. In this case, the likelihood can be computed exactly by the means of the change of variable formula, and latent variables can be inferred deterministically. A variety of different normalizing flows has recently been proposed, see *e.g.* (Rezende and Mohamed, 2015; Kingma et al., 2016; Dinh et al., 2017; Chen et al., 2018; Kingma and Dhariwal, 2018; Behrmann et al., 2019; Grathwohl et al., 2019). See (Kobyzev et al., 2019; Papamakarios et al., 2019) for recent comprehensive reviews on normalizing flows.

Affine coupling layers (Dinh et al., 2017) allow for a computation of the inverse in a closed form that is as efficient to compute as the function itself. Within this approach, the activations A^ℓ in layer ℓ are partitioned in two groups, A_1^ℓ and A_2^ℓ. The first group is unchanged, and used to update the other group by scaling and translation, *i.e.* $A_2^{\ell+1} = A_2^\ell \odot s(A_1^\ell) + t(A_1^\ell)$, where $\odot$ denotes element-wise multiplication, and $s(\cdot)$ and $t(\cdot)$ can be arbitrary (non-invertible) neural networks. The inverse is trivially obtained by subtraction and division, since $A_1^{\ell+1} = A_1^\ell$. Many coupling layers with changing variable partitioning can be stacked to construct a complex invertible flow. Training and sampling the model require to compute the flow reverse directions, and since affine coupling layers are equally efficient in both directions, it means that both processes are fast. This is in contrast to some other normalizing flows, such as invertible ResNets (Chen et al., 2018; Behrmann et al., 2019), or planar and radial flows (Rezende and Mohamed, 2015), for which the inverse flow exists, but does not have a closed form.

Neural ordinary differential equations were recently proposed as a generalization of deep residual networks (ResNets, (He et al., 2016a,b)) in the limit of infinite depth (Chen et al., 2018). In a ResNet the activations $A^{\ell+1}$ in a layer $\ell+1$ are defined as the sum of the activations A^ℓ of the previous layer plus a residual: $A^{\ell+1} = A^\ell + f_\ell(A^\ell)$, where $f_\ell(\cdot)$ consists of a few non-linear layers. In the Neural-ODE the residual is interpreted as a continuous time dynamic $\partial A^\ell/(\partial \ell) = f(A^\ell, \ell)$. Standard ODE solvers can be used to perform the propagation both

forwards and backwards in time. The authors of (Chen et al., 2018) demonstrated that Neural ODEs can be used to define normalizing flow, which are referred to as "continuous normalizing flows".

Several conditional flow-based models have recently been proposed for vision tasks. In (Lu and Huang, 2020) flows conditioned on an input image are used for image segmentation, inpainting, denoising, and depth refinement. Their model is trained directly via maximum likelihood estimation, as their model does not include a global latent variable. Similarly, C-Flow (Pumarola et al., 2020) does not involve a global latent variable, and rather than treating point clouds as sets, it sorts the points to a regular pixel grid, and applies 2D normalizing flows for single image point cloud reconstruction. A conditional VAE model, where flow is used to define a flexible distribution on the latent variable given the conditioning data was introduced by (Bhattacharyya et al., 2019). This is similar in structure to our model for single view reconstruction. Their experiments, however, concern the prediction of point trajectories in 2D for hand-written digits, and traffic participants such as pedestrians and cars. The generative image model of (Lucas et al., 2019) is related to ours, as a VAE model with a flow-based decoder. The application contexts, RGB images *vs*. point clouds, and resulting architectures are, however, quite different.

Point Cloud Generating Networks. Deep learning models for point cloud processing have received significant attention in recent years, see *e.g*. (Qi et al., 2017a; Klokov and Lempitsky, 2017; Qi et al., 2017b; Zaheer et al., 2017; Achlioptas et al., 2018; Groueix et al., 2018; Su et al., 2018; Li et al., 2018a; Yang et al., 2019). The PointNet architecture (Qi et al., 2017a) was the first to propose a deep network for recognition of point clouds. The points are first processed in an identical and independent manner by an MLP, and global max-pooling is used to aggregate the per-point information. The global features are further processed by a second MLP, and either directly used for classification, or appended to the local features for semantic segmentation. KD-Net (Klokov and Lempitsky, 2017) and PointNet++ (Qi et al., 2017b) add a notion of spatial proximity to the architecture, replacing global max-pooling with local aggregation. While these models can interpret point clouds, they cannot generate them.

Early point cloud generating networks (Fan et al., 2017; Achlioptas et al., 2018) produce point clouds with a fixed number of points n, by using a network with $n \times 3$ outputs. AtlasNet (Groueix et al., 2018) mitigates this limitation by using a set of k square 2D patches, and deforming each of these non-linearly by using k patch-specific MLPs that takes as input 2D patch coordinates as well as a global shape representation. By sampling more points from the 2D patches, a denser 3D point cloud is obtained. The shape vector is obtained from a point cloud encoder network (for autoencoding), or from a CNN trained for single-view image reconstruction. The point cloud GAN (PC-GAN, (Li et al., 2018a)) is related, but uses a single generator that takes a global shape vector as input together with (arbitrarily many) samples from a unit Gaussian. Similarly to (Fan et al., 2017), AtlasNet is a conditional model, that generates point clouds *given* another point cloud or an image. In contrast, PC-GAN includes a second generator that models a distribution on the latent shape space, so it can generate point clouds in an unconditional manner.

The high-level hierarchical latent variable structure of PointFlow (Yang et al., 2019) is similar to PC-GAN. Rather than using adversarial training, however, they train the model using a VAE-like approach in which an inference network produces an approximate posterior on the latent shape representation. Moreover, they use continuous normalizing flows (Chen et al., 2018) to define a prior on the shape space, and a conditional distribution on 3D points given a latent shape representation. Our work is based on the same high-level VAE-like structure as

PointFlow, but differs in the design of the network components. Most importantly, we make use of efficient "discrete" affine coupling layers, avoiding the use of computationally expensive ODE solvers for training and generation needed for the "continuous" flows, resulting in a significant speed-up to train and sample from the model. We describe our "Discrete Point Flow Networks" in the next section.

Despite their mathematical elegance, it is not clear whether continuous normalizing flows are to be preferred over their discrete counterparts in terms of computational efficiency and modeling capabilities. We explore a model similar to PointFlow, but build upon computationally much more efficient discrete normalizing flows with affine coupling layers.

4.3 Discrete Point Flow Networks

In this section we first present the high-level hierarchical latent variable model, followed by a more detailed description of the model components in Section 4.3.2.

4.3.1 Hierarchical Latent Variable Model for Point Cloud Generation

Our goal is to define a generative model over point clouds of variable size that represent 3D shapes. The defining characteristics of point clouds are that the number of points may vary from one cloud to another, and that there is no inherent ordering among the points.

Let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be a point cloud with $x_i \in \mathbb{R}^d$, where $d = 3$ for point clouds for 3D shapes. The dimension d may be larger in some cases, *e.g.* $d = 6$ when modeling 3D points equipped with surface normals. An exchangeable distribution is one that is invariant to permutations of the data, *i.e.*

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_n) = p(\mathbf{x}_{\pi_1}, \ldots, \mathbf{x}_{\pi_n}), \tag{4.1}$$

where π is a permutation of the integers $1, \ldots, n$. Note that independence implies exchangeability, but the reverse does not hold.

De Finetti's representation theorem (Finetti, 1937) states that any exchangeable distribution can be written as a factored distribution, conditioned on a latent variable:

$$p(X) = \int p_\psi(\mathbf{z}) \prod_{\mathbf{x} \in X} p_\theta(\mathbf{x}|\mathbf{z}) \mathrm{d}\mathbf{z}. \tag{4.2}$$

In the case of 3D point cloud modeling, the latent variable $\mathbf{z}$ can be thought of as an element in an abstract shape space, sampled from a prior $p_\psi(\mathbf{z})$. This construction allows for point clouds of different cardinality, since conditioned on the shape representation $\mathbf{z}$, the elements of the point cloud are sampled i.i.d. Given this general framework, also adopted in (Li et al., 2018a; Yang et al., 2019), the challenge is to:

1. Design a flexible model so that the conditional distribution $p_\theta(\mathbf{x}|\mathbf{z})$ concentrates around the surface of the object represented by $\mathbf{z}$.

2. Mitigate the intractability of the integral in Equation (4.2) during training when using, *e.g.*, deep neural networks to construct $p_\theta(\mathbf{x}|\mathbf{z})$.

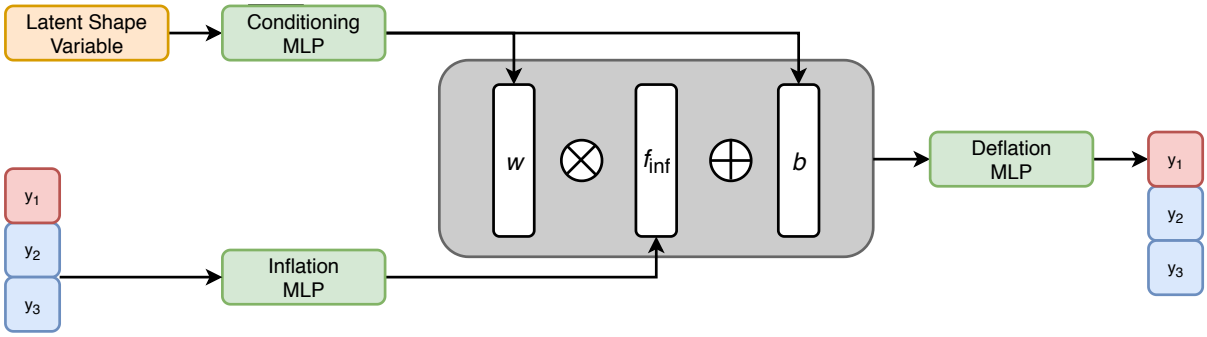

Figure 4.2: Architecture of our conditional affine coupling layer applied to a single 3D point, with red dimension of the point being updated given the blue ones.

Before we consider the design of $p_\theta(\mathbf{x}|\mathbf{z})$ and $p_\psi(\mathbf{z})$ in Section 4.3.2, we describe how to deal with the integral in Equation (4.2) using the VAE framework (Kingma and Welling, 2014).

We efficiently approximate the intractable posterior $p(\mathbf{z}|X)$ with an amortized inference network $q_\phi(\mathbf{z}|X)$. The approximate posterior allows us to define a variational bound on the likelihood in Equation (4.2) using Jensen's inequality (Bishop, 2006):

$$\ln p(X) \;\geq\; \sum_{\mathbf{x}\in X} \mathbb{E}_{q_\phi(\mathbf{z}|X)}[\ln p_\theta(\mathbf{x}|\mathbf{z})] - \mathcal{D}_{\mathrm{KL}}(q_\phi(\mathbf{z}|X)||p_\psi(\mathbf{z})) \equiv -\mathcal{F}. \tag{4.3}$$

The first term aims to reconstruct the points $\mathbf{x} \in X$ using shape representations sampled from $q_\phi(\mathbf{z}|X)$, whereas the second term ensures that the approximate posterior cannot arbitrarily deviate from the prior. Following (Kingma and Welling, 2014; Rezende et al., 2014) we use Monte Carlo sampling and the reparametrization trick to jointly minimize the loss $\mathcal{F}$ over θ, ψ and ϕ using stochastic gradient descent. The distributions $q_\phi(\mathbf{z}|X)$, $p_\theta(\mathbf{x}|\mathbf{z})$, and $p_\psi(\mathbf{z})$ and the underlying network architectures that make up the loss are detailed in the following section.

4.3.2 Design of Model Components

Shape-conditional Point Distribution. The density on points for a given latent shape, $p_\theta(\mathbf{x}|\mathbf{z})$, needs to be flexible enough to concentrate its support around the surface of the 3D shape. To this end we construct a conditional form of normalizing flows based on affine coupling layers (Dinh et al., 2017).

Let $\mathbf{y} \in \mathbb{R}^3$ denote a latent variable for each 3D point $\mathbf{x}$, with a Gaussian conditional distribution given by $p_\theta(\mathbf{y}|\mathbf{z}) = \mathcal{N}(\mathbf{y}; \nu_\theta(\mathbf{z}), \mathrm{diag}(\omega_\theta(\mathbf{z})))$, where $\nu_\theta(\mathbf{z})$ and $\omega_\theta(\mathbf{z})$ are non-linear functions of $\mathbf{z}$. In the affine coupling layer, we partition the coordinates of $\mathbf{y}$ in two groups, $\mathbf{y}^c$ and $\mathbf{y}^u$, and update $\mathbf{y}^u$ by affine transformation conditioned on $\mathbf{y}^c$ and the latent shape representation $\mathbf{z}$, *i.e.* $\mathbf{x}^u = \mathbf{y}^u \odot s_\theta(\mathbf{y}^c, \mathbf{z}) + t_\theta(\mathbf{y}^c, \mathbf{z})$, while leaving the conditioning coordinates unchanged, *i.e.* $\mathbf{x}^c = \mathbf{y}^c$. To achieve the desired expressivity, we stack many affine coupling layers, cycling through the six possible partitionings of the three coordinates. Each coupling layer in the resulting flow $f_\theta(\mathbf{x}; \mathbf{z})$ is conditioned on the latent shape representation $\mathbf{z}$ by the means of the FiLM conditioning mechanism (Perez et al., 2018) in the scaling and translation functions.

In practice, the scaling and translation functions are implemented by MLPs, which inflate the dimensionality of $\mathbf{y}^c$ to D_{inf}, and then deflate it to the dimensionality of $\mathbf{y}^u$. Simultaneously, a separate MLP takes the latent variable $\mathbf{z}$ and outputs conditioning coefficients of size D_{inf}, with

which we multiply and shift the inflated hidden units in the scaling and translation functions. In Figure 4.2 we provide an overview of the architecture of our conditional coupling layers.

Using $f_\theta(\mathbf{x};\mathbf{z})$ to denote the invertible flow network that maps $\mathbf{x}$ to $\mathbf{y}$, the change of variable formula allows to write the density of 3D points $\mathbf{x}$ given $\mathbf{z}$ as:

$$p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}\left(f_\theta(\mathbf{x};\mathbf{z}); \nu_\theta(\mathbf{z}), \operatorname{diag}\left(\omega_\theta(\mathbf{z})\right)\right) \left|\det\left(\frac{\partial f_\theta(\mathbf{x};\mathbf{z})}{\partial \mathbf{x}^\top}\right)\right|, \tag{4.4}$$

which enters into the loss defined in Equation (4.3).

Amortized Inference Network. The amortized inference network $q_\phi(\mathbf{z}|X)$ takes a point cloud and produces a distribution on the latent shape representation. We use a permutation invariant design based on the PointNet architecture for shape classification (Qi et al., 2017a). As an output, the model produces the mean and diagonal covariance matrix of a Gaussian on $\mathbf{z} \in \mathbb{R}^D$, *i.e.* $q_\phi(\mathbf{z}|X) = \mathcal{N}\left(\mathbf{z}; \mu_\phi(X), \operatorname{diag}\left(\sigma_\phi(X)\right)\right)$.

Latent Shape Prior. Rather than using a unit Gaussian prior in the latent space, as is common in deep generative models, we learn a more expressive prior $p_\psi(\mathbf{z})$ by means of another normalizing flow $g_\psi(\mathbf{z})$ based on affine coupling layers, similar to (Chen et al., 2017; Yang et al., 2019). In our experiments it reduces the KL divergence in Equation (4.3) by adapting the prior to fit the marginal posterior $\sum_X q_\phi(\mathbf{z}|X)$, rather than forcing the inference network to induce a unit Gaussian marginal posterior, resulting in improved generative performance. Using this construction, we obtain the KL divergence as:

$$\mathcal{D}_{\mathrm{KL}}(q_\phi(\mathbf{z}|X)||p_\psi(\mathbf{z})) = \mathbb{E}_{q_\phi(\mathbf{z}|X)}[\ln p_\psi(\mathbf{z})] - \mathcal{H}(q_\phi(\mathbf{z}|X)) \tag{4.5}$$

$$= \mathbb{E}_{q_\phi(\mathbf{z}|X)}\left[\ln \mathcal{N}\left(g_\psi(\mathbf{z}); \eta, \operatorname{diag}\left(\kappa\right)\right) + \ln\left|\det\left(\frac{\partial g_\psi(\mathbf{z})}{\partial \mathbf{z}^\top}\right)\right|\right] - 1^\top \ln \sigma_\phi(X), \tag{4.6}$$

where we use Monte Carlo sampling to approximate the expectation.

Single-View Reconstruction Architecture. For single-view reconstruction, we follow (Klokov et al., 2019) and define the model as:

$$p(X|\mathbf{v}) = \int p_\psi(\mathbf{z}|\mathbf{v}) \prod_{\mathbf{x}\in X} p_\theta(\mathbf{x}|\mathbf{z})\mathrm{d}\mathbf{z}, \tag{4.7}$$

where we replaced the latent shape prior $p_\psi(\mathbf{z})$ with an image conditioned one, $p_\psi(\mathbf{z}|\mathbf{v})$. In this case, the latent shape flow g_ψ does not deform a parametric Gaussian, but rather a Gaussian whose mean and variance are computed from an image $\mathbf{v}$ by a CNN encoder. We train the model by optimizing a variational bound similar to Equation (4.3), and using the PointNet inference network to obtain an approximate posterior. Figure 4.3 provides an overview of the data flow between the model components for training and point cloud generation.

4.4 Experiments

We first describe our experimental setup in Section 4.4.1, and then present our experimental results for point cloud autoencoding, generation, and single-view reconstruction.

4.4.1 Data and Metrics

Datasets. In order to provide a comparison with prior academic studies on shape generation, we perform experiments on subsets of ShapeNet (Chang et al., 2015) dataset. For autoencoding

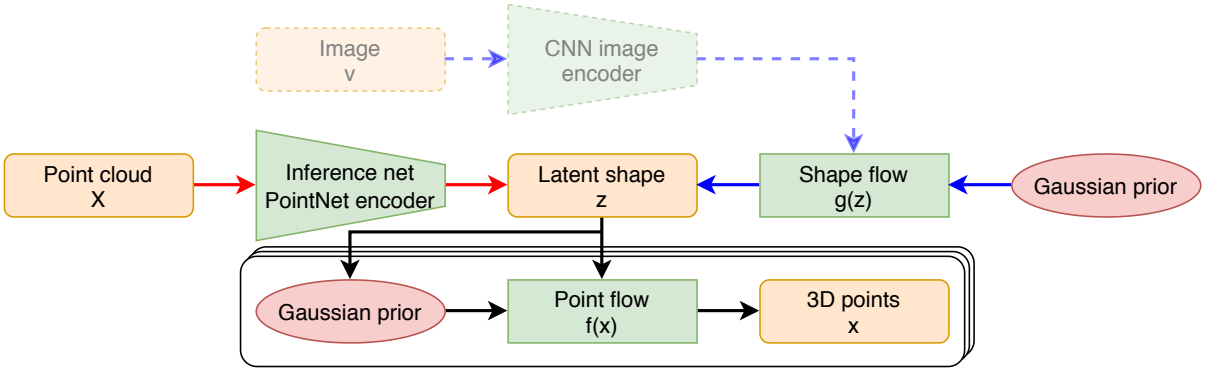

Figure 4.3: Overview of DPF-Net: arrows indicate data flow to sample new point clouds (blue) and point cloud autoencoding (red), black arrows are used in both processes. During training flow modules are traversed in the reverse direction. For single-view reconstruction the shape prior is conditioned on the image (dashed).

we use the ShapeNetCore.v2, containing roughly 55k meshes from 55 classes. In the generative setting, following (Yang et al., 2019), we use single class subsets (*airplanes*, *cars*, and *chairs*) from the same dataset. For the single-view reconstruction task we used a subset of 13 major classes of ShapeNetCore.v1 from (Choy et al., 2016), which comes with rendered 137×137 images from 24 randomized viewpoints per shape. We substitute the voxel grids provided by Choy *et al*. with the original meshes to sample point clouds for training and evaluation.

Data Split. For autoencoding we use a random split of the data, distributed across train, validation, and test sets in a 70/10/20 proportion per class. In the generative setting, we use single class subsets from the same random split. By using a random data split per class we intentionally deviate from the official ShapeNet data split, used in (Yang et al., 2019), which splits into significantly different data subsets per class. For example, in case of airplanes, training and validation sets mostly contain regular passenger aircraft, while the test set is populated with fighter jets and spaceships. While such a split could be useful in the context of autoencoding to assess out-of-distribution generalization, a significant mismatch between training and test set is undesirable for evaluation of generative models which are supposed to fit the training distribution. For single-view reconstruction we use the train/test split from Choy et al. (2016).

Normalization. The original meshes in the ShapeNet are not normalized for position and scale which negatively affects the reconstruction quality. We therefore, additionally use a normalized version of the dataset, where we preprocess each mesh separately so that the sampled point clouds are (approximately) zero mean, and tightly fit in a unit diameter sphere. For generative experiments we use models trained and evaluated on normalized data. In case of the autoencoding, we report results for two separate DPF-Nets trained either on non-normalized or normalized data, where in the latter case we rescale point clouds to original scales before evaluation for comparability with the rest of the models. While using non-normalized data, similarly to (Yang et al., 2019), we perform global normalization across all shapes by translation to the aggregate center of all the training shapes, but do not rescale to unit global variance. For single-view reconstruction we compare models trained on normalized data, but evaluate them in the unit radius sphere scale for comparability with related work.

Point clouds are uniformly sampled from the meshes by sampling polygons with a probability proportional to their area, and then uniformly sampling a point per each selected polygon. Unlike previous works using precomputed point clouds, we perform this procedure on the fly,

thus obtaining a different random point cloud each time we process a 3D shape. During both training and evaluation we sample two point clouds for each shape: one is used as input to the inference network, the other for optimization or evaluation of the decoder. We use $2,048$, $2,048$, and $2,500$ points for training and quantitative evaluation for generative, autoencoding, and single-view reconstruction tasks accordingly.

Evaluation Metrics. We follow the standard protocol (Achlioptas et al., 2018; Fan et al., 2017; Yang et al., 2019) and use Chamfer distance (CD) and earth mover's distance (EMD) to assess point cloud reconstructions. To measure the generative properties we follow (Achlioptas et al., 2018; Yang et al., 2019), and use metrics to compare equally sized *sets* of generated and reference point clouds:

- The Jensen-Shannon divergence (JSD) compares the marginal distributions obtained by taking the union of all generated (or reference) point clouds, and quantizing the distributions to a voxel grid.
- The Minimum matching distance (MMD) computes the average distance of reference point clouds to their nearest (in CD/EMD) generated point cloud.
- Coverage (COV) is the fraction of reference point clouds matched by minimum CD/EMD distance by at least one generated point cloud.
- 1-nearest neighbour accuracy (1-NNA) classifies generated and reference point clouds as belonging to either of these two sets using a leave-one-out 1-nearest neighbor classifier (using CD/EMD). Ideal accuracy is 50%.

For single-view reconstruction we additionally report F1-score. For more detailed descriptions of these metrics we refer to (Achlioptas et al., 2018; Yang et al., 2019).

4.4.2 Experimental Setup

Inference Network. We use a PointNet encoder (Qi et al., 2017a) with the number of features progressing over layers as $3-64-128-256-512$, followed by a max-pooling across points, and two fully-connected layers of sizes 512 and D, where D is the size of the latent space. We use $D=512$ in the autoencoding and single-view reconstruction experiments, and $D=128$ for generative modeling.

Latent Shape Prior. We use 14 affine coupling layers to construct the latent space prior. In the coupling layers, we alternate between two orthogonal partitioning schemes: odd and even dimensions are split in different groups; the first $D/2$ dimension go in one group, and the remaining in the other. For single-view reconstruction we use ResNet18 image encoder, similarly to (Groueix et al., 2018; Wang et al., 2018, 2019a).

Point Decoder. The point decoder $p(\mathbf{x}|\mathbf{z})$ starts with a three-dimensional Gaussian with mean and variances computed from $\mathbf{z}$ by an MLP with two hidden layers. This Gaussian is transformed by 63 of our conditioned affine coupling layers, each consisting of (i) two fully-connected layers that map z to the FiLM conditioning coefficients, each of size 64, (ii) two fully-connected layers that inflate input dimensionality to 64 hidden units (at which point the FiLM conditioning is performed), (iii) a final fully-connected layer that deflate the dimensionality to compute the scaling and translation functions.

Model	Nr. params., 10^6	Mem. footprint, Mb/sample	tr. time, ms/sample	total tr. time, days	gen. time, ms/sample
PointFlow (Yang et al., 2019)	1.63	470	500	80	150
DPF-Net (Ours)	3.76	370	16	1.1	4

Table 4.1: Efficiency comparison for DPF-Nets and PointFlow generative models.

Baselines. We retrained AtlasNet (Groueix et al., 2018), l-GAN-CD/EMD (Achlioptas et al., 2018), PointFlow (Yang et al., 2019), and DCG (Wang et al., 2019a) with our split of the ShapeNet dataset, using the implementation provided by the authors and our data processing pipeline. For improved comparability we also modified the point cloud encoders in all models to match each other (except for l-GANs, since it significantly worsened their results). To match other approaches, we used $2,048$ points per cloud for AtlasNet in the autoencoding task and consequently set the number of learned primitives to 16 to keep the property of sampling the equal number of points from every primitive, (originally it uses 25 primitives and samples $2,500$ points and we kept these settings for single-view reconstruction evaluations).

Oracles. For all the tasks we also provide an "oracle" to assess the best possible performance values. For autoencoding and single-view reconstruction, the oracle samples a second point cloud from the ground truth mesh, rather than generating a point cloud. For generative modeling, the oracle uses the point clouds from the training set, instead of sampling point clouds from the model.

Training Details. All the models were trained with AMSGrad optimizer (Reddi et al., 2018) with decoupled weight decay regularization (Loshchilov and Hutter, 2019) with step-like schedule for the learning rate. All the hyperparameters for all the experiments can be found on our project webpage.[1]

4.4.3 Generative Modeling Evaluation

Efficiency Comparison with PointFlow. We compare our DPF-Networks in terms of computational efficiency and memory footprint to PointFlow (Yang et al., 2019). We train both models for point cloud generation, and report the number of parameters and total training time. To compute the training memory footprint, training time, and generation time per sample (point cloud), we divided total GPU memory occupied during training, batch iteration time, and batch generation time, respectively, by the batch size.

Both models were run on a single TITAN RTX GPU. We estimate the total training time for PointFlow after the initial 100 epochs of training, which took 2 days, and assume that the full training procedure requires $4,000$ epochs, as reported by the authors of PointFlow. We observed that the training procedure slowed down over the course of training, because ODE-solver gradually increases the number of iterations to meet the required tolerance. Thus, all timings in Table 4.1 for PointFlow should be understood as lower bounds.

From the results in Table 4.1 we see that even though DPF-Networks have more parameters, the associated training memory footprint is lower and, our model is approximately 30 times faster both in training and inference iterations, and can be trained in a single day.

Quantitative Results. We compare to l-GANs and PointFlow models and report oracle performance as a reference. Given the prohibitive computation cost of complete PointFlow

[1] https://github.com/Regenerator/dpf-nets

Category	Model	JSD↓	MMD↓		COV↑, %		1-NNA↓, %	
			CD	EMD	CD	EMD	CD	EMD
Airplane	l-GAN-CD [1]	2.76 ± 0.16	**5.69** ± 0.04	5.16 ± 0.02	39.5 ± 0.8	17.1 ± 0.6	72.9 ± 0.8	92.1 ± 0.6
	l-GAN-EMD [1]	1.77 ± 0.13	6.05 ± 0.04	**4.15** ± 0.02	39.7 ± 1.4	40.4 ± 1.2	75.7 ± 0.6	73.0 ± 1.2
	PointFlow [2]	1.42 ± 0.12	6.05 ± 0.05	4.32 ± 0.01	**44.7** ± 1.2	**48.4** ± 1.0	**70.9** ± 1.0	**68.4** ± 1.0
	DPF-Nets	**0.94** ± 0.11	6.07 ± 0.04	4.26 ± 0.02	**46.8** ± 1.2	**48.4** ± 0.9	**70.6** ± 1.0	**67.0** ± 1.2
	Oracle	0.50 ± 0.04	5.97 ± 0.09	3.98 ± 0.01	51.4 ± 1.0	52.7 ± 1.3	49.8 ± 1.3	48.2 ± 1.1
Car	l-GAN-CD [1]	2.65 ± 0.07	**8.83** ± 0.06	5.36 ± 0.01	41.3 ± 0.8	15.9 ± 1.3	**62.6** ± 0.6	92.7 ± 0.4
	l-GAN-EMD [1]	1.31 ± 0.10	9.00 ± 0.08	**4.40** ± 0.01	38.3 ± 1.2	32.9 ± 0.7	65.2 ± 0.4	**63.2** ± 1.0
	PointFlow [2]	0.59 ± 0.02	9.53 ± 0.06	4.71 ± 0.01	**42.3** ± 1.0	35.8 ± 1.3	70.1 ± 0.9	74.2 ± 0.6
	DPF-Nets	**0.45** ± 0.02	9.59 ± 0.04	4.61 ± 0.01	**43.4** ± 0.9	**45.8** ± 1.0	70.3 ± 0.6	**64.3** ± 1.5
	Oracle	0.37 ± 0.03	9.24 ± 0.06	4.56 ± 0.01	52.8 ± 1.1	52.7 ± 0.9	50.9 ± 1.1	50.5 ± 1.2
Chair	l-GAN-CD [1]	3.65 ± 0.09	**16.66** ± 0.08	7.91 ± 0.02	42.3 ± 0.5	17.1 ± 0.5	68.5 ± 0.5	96.5 ± 0.1
	l-GAN-EMD [1]	1.27 ± 0.06	**16.78** ± 0.07	**5.75** ± 0.01	44.3 ± 0.9	43.8 ± 1.0	66.6 ± 0.6	67.8 ± 0.7
	PointFlow [2]	1.51 ± 0.11	17.15 ± 0.10	6.20 ± 0.01	43.3 ± 0.8	**46.5** ± 1.0	67.0 ± 0.3	70.4 ± 0.6
	DPF-Nets	**1.01** ± 0.06	17.08 ± 0.11	6.14 ± 0.01	**46.9** ± 0.8	**48.5** ± 1.1	**63.5** ± 1.3	**64.8** ± 0.7
	Oracle	0.49 ± 0.01	16.39 ± 0.07	5.71 ± 0.01	52.8 ± 0.8	53.4 ± 1.1	49.7 ± 0.7	49.6 ± 0.9

Table 4.2: Generative modeling results. Oracle results are underlined when they are not the best. JSD and MMD-EMD are multiplied by 10^2, MMD-CD by 10^4.

training, we provide results obtained after training for four days, which is four times the full training time of DPF-Net in the same setting. In order to account for random sampling every model is evaluated using ten different sets of generated objects, each of the size of the test set. Thus, for each metric we report mean values and standard deviations calculated over ten runs. In addition to the best values, in Table 4.2 we also write in bold results that are within two standard deviations of the best result.

Overall, DPF-Networks yield the best results in terms of JSD, COV-CD/EMD and 1-NNA-CD/EMD, except for the 1-NNA-CD for *car*. This confirms that our DPF-Network is capable of generating more realistic and diverse sets of point clouds, for random samples from our model see Figure 4.1.

L-GAN-CD experiences mode collapses, and generates objects with good CD values, but with very poor coverage and 1-NNA in terms of the EMD metric. PointFlow shows performance similar to ours, except for JSD, while being significantly slower in both training and sampling. In contrast to the evaluations performed in (Yang et al., 2019), based on the official split, in our experiments the oracle obtains the best performances for all metrics, except for MMD (see underlined results). We believe that this highlights the fact the MMD metric does not favor diversity in the generated point clouds, but instead favors point clouds with low CD/EMD distances to all the reference shapes. If the generated point clouds contain a subset of high quality modes from the test subset, the metric can yield good results, even better than the oracle. DPF-Nets and PointFlow yield qualitatively similar point cloud samples, we provide a comparison of samples in Figure 4.4.

Qualitative Results. We additionally conducted several experiments to qualitatively evaluate DPF-Networks. We provide a qualitative comparison of our per-class generative models with the corresponding PointFlow models in Figure 4.4. Both models are able to generate various detailed class-specific samples, however, the probabilistic nature of both models encourages them to produce increased amounts of noise in the parts where they are least confident with predictions.

In Figure 4.5 we provide latent space interpolations from the models trained for the Airplane, Car, and Chair classes. We sample two shapes from the test data (left- and right-most in the

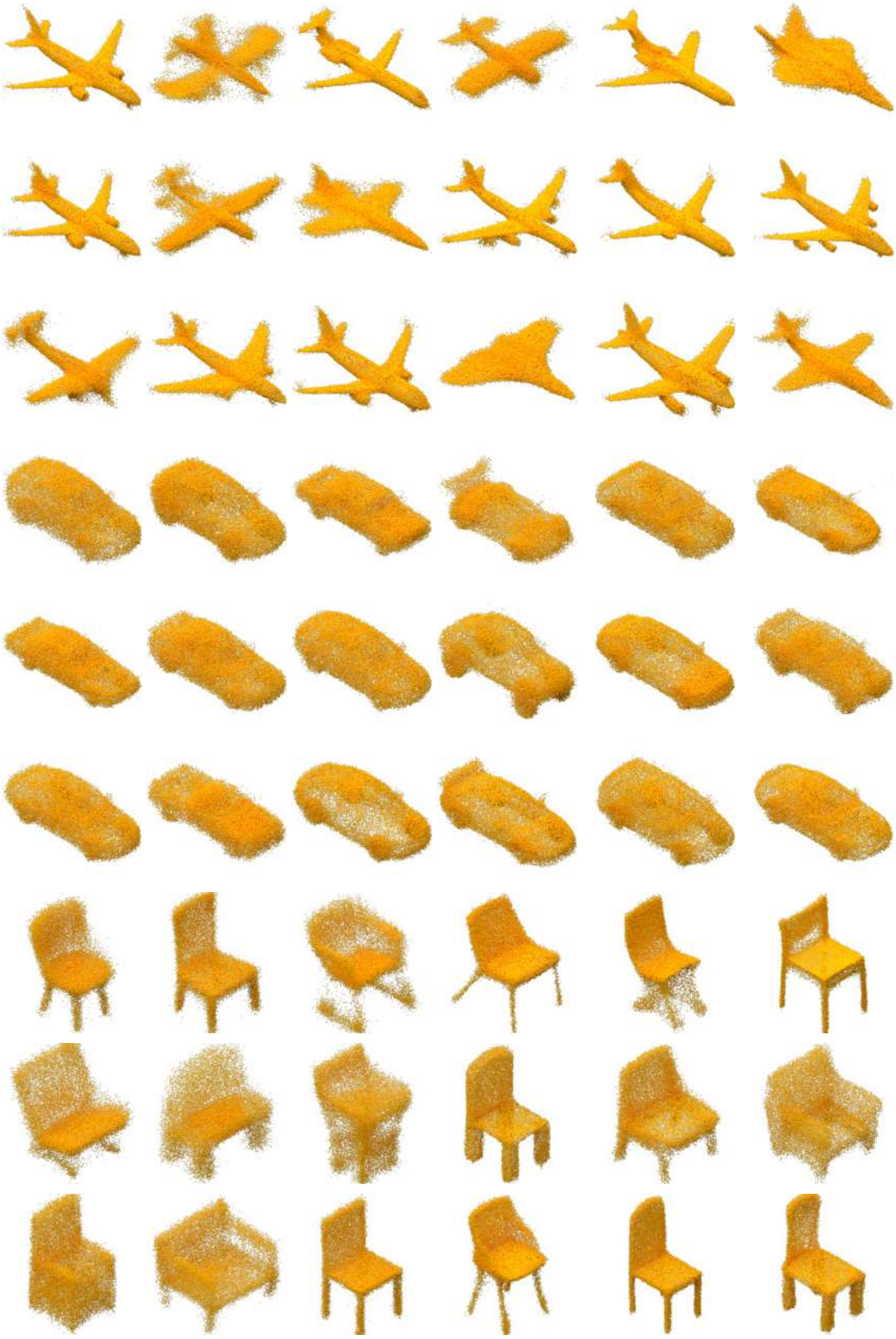

Figure 4.4: Random samples generated by PointFlow (Columns 1—3) and DPF-Network (Columns 4—6). For each sample we generated $32,768$ points.

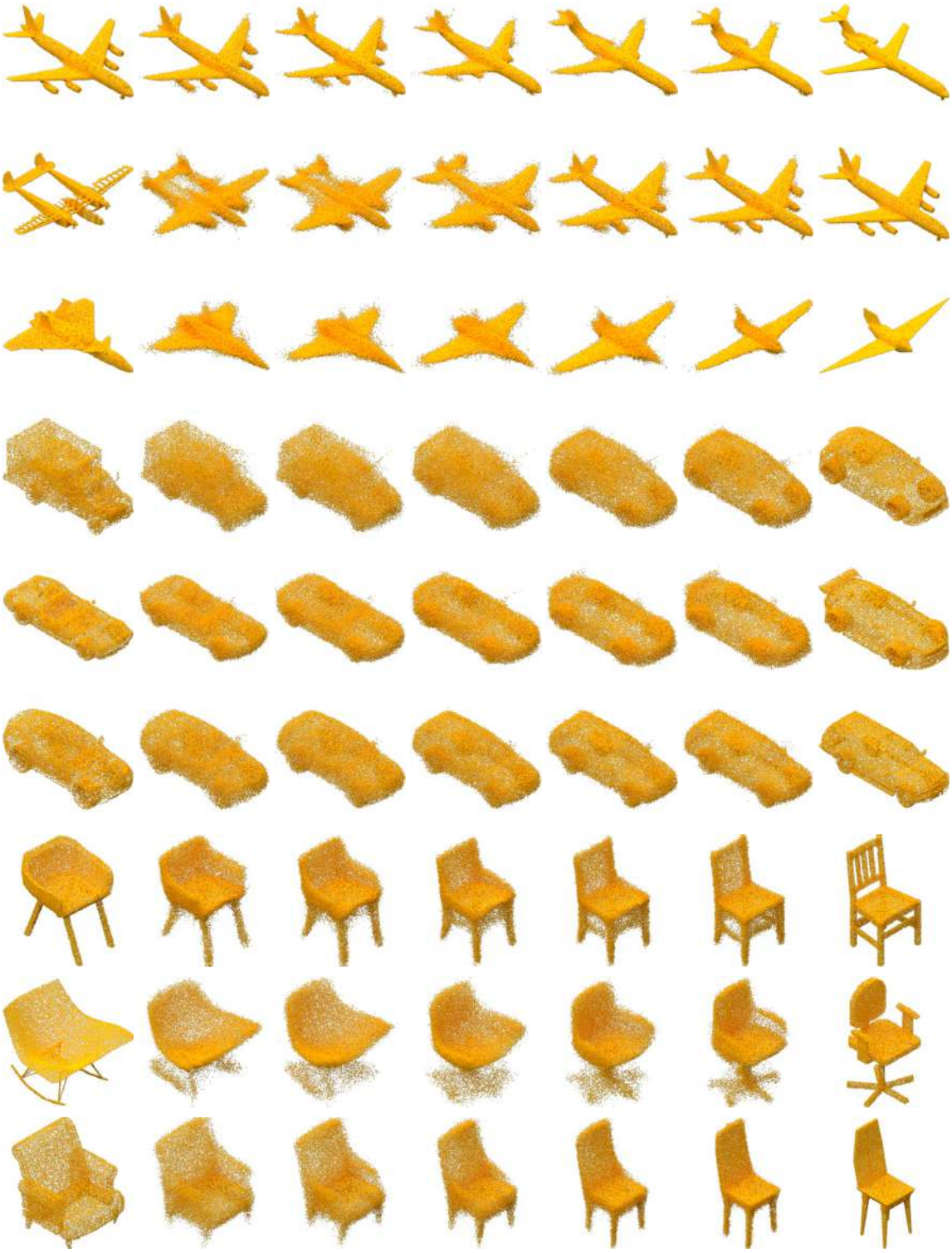

Figure 4.5: Generated samples from latent space interpolations between two ground truth samples obtained with DPF-Networks. Left and rightmost columns are ground truth samples, each shape between them is a reconstruction from a latent shape obtained as a differently weighted mean of corresponding latent shapes. For each sample we generated $32,768$ points.

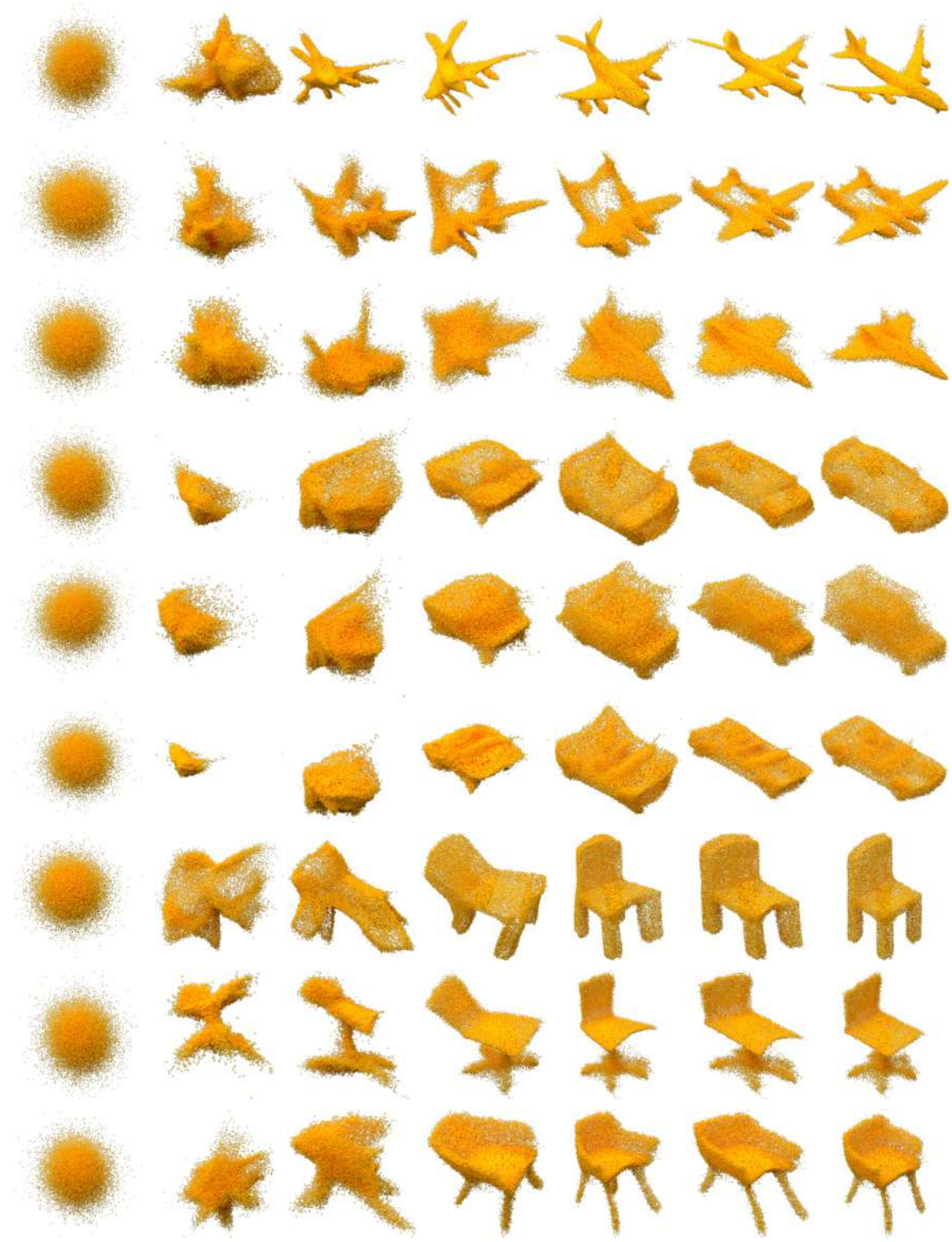

Figure 4.6: Evolution of points in reconstructed samples across different layers of the flow in DPF-Networks. For each sample we generated $32,768$ points.

Metric	CD $\times 10^4$	EMD $\times 10^2$
l-GAN-CD (Achlioptas et al., 2018)	7.07	7.70
l-GAN-EMD (Achlioptas et al., 2018)	9.18	5.30
AtlasNet (Groueix et al., 2018)	**5.66**	5.81
PointFlow† (Yang et al., 2019)	7.54	5.18
PointFlow*	10.22	6.58
DPF-Net, orig.	6.85	5.06
DPF-Net, norm.	6.17	**4.37**
Oracle	3.10	3.13

Table 4.3: Autoencoding results. † - results from (Yang et al., 2019) on the official split, $*$ - results for equal training time as DPF-Net on the random split.

figure), and interpolate between corresponding latent variable samples to obtain a path in the shape space. The model is able to produce sensible smooth samples from the interpolated latent codes, which demonstrates that it is able to learn a rich latent shape space, capturing various shape geometry features.

In Figure 4.6 we provide examples of the generating flow for sampled point clouds from the class-specific DPF-Networks. We sample a shape from data, obtain initial Gaussian from the corresponding latent variable, sample $32,768$ points by the flow, and then visualize the evolution of the point cloud across the initial Gaussian and layers 32, 48, 56, 60, 62, 63 of the generative flow network.

4.4.4 Autoencoding Evaluation

We compare DPF-networks with other models in terms of autoencoding performance in Table 4.3. Similarly to generative experiments, we restricted the training time of PointFlow, this time, to match the training time of our approach which was approximately a week. Among models trained on non-normalized data, DPF-Net (orig.) achieve the best results in the EMD metric and second best in the CD metric, outperformed only by the non-generative AtlasNet which is trained by optimization of the CD metric. The DPF-Net outperforms both l-GANs which were specifically optimized for the CD/EMD metrics, while being trained by optimization of the likelihood lower bound. Importantly DPF-Nets outperform PointFlow in both metrics under the same and extended computational budget.

When our model is trained on normalized data, results significantly improve, achieving state-of-the-art among generative models for both metrics. This underlines the importance of proper data normalization for shape modeling.

In Figure 4.7 and Figure 4.8 we qualitatively compare our autoencoding results to l-GANs, AtlasNet, and PointFlow. All approaches can work with arbitrary size inputs, but only AtlasNet, PointFlow, and DPF-Nets can reconstruct with arbitrary density. In this comparison we use 512 and $32,768$ points as sparse and dense inputs, while reconstructing fixed $2,048$ points for l-GANs and $32,768$ point for AtlasNet, PointFlow and DPF-Nets. Models with better CD values (l-GAN-CD, AtlasNet) tend to concentrate points in some regions of reconstructed shapes, while models with better EMD values (l-GAN-EMD, DPF-Nets) distribute points more evenly. While AtlasNet achieves best CD, its reconstructions contain sharp plane-like artifacts.

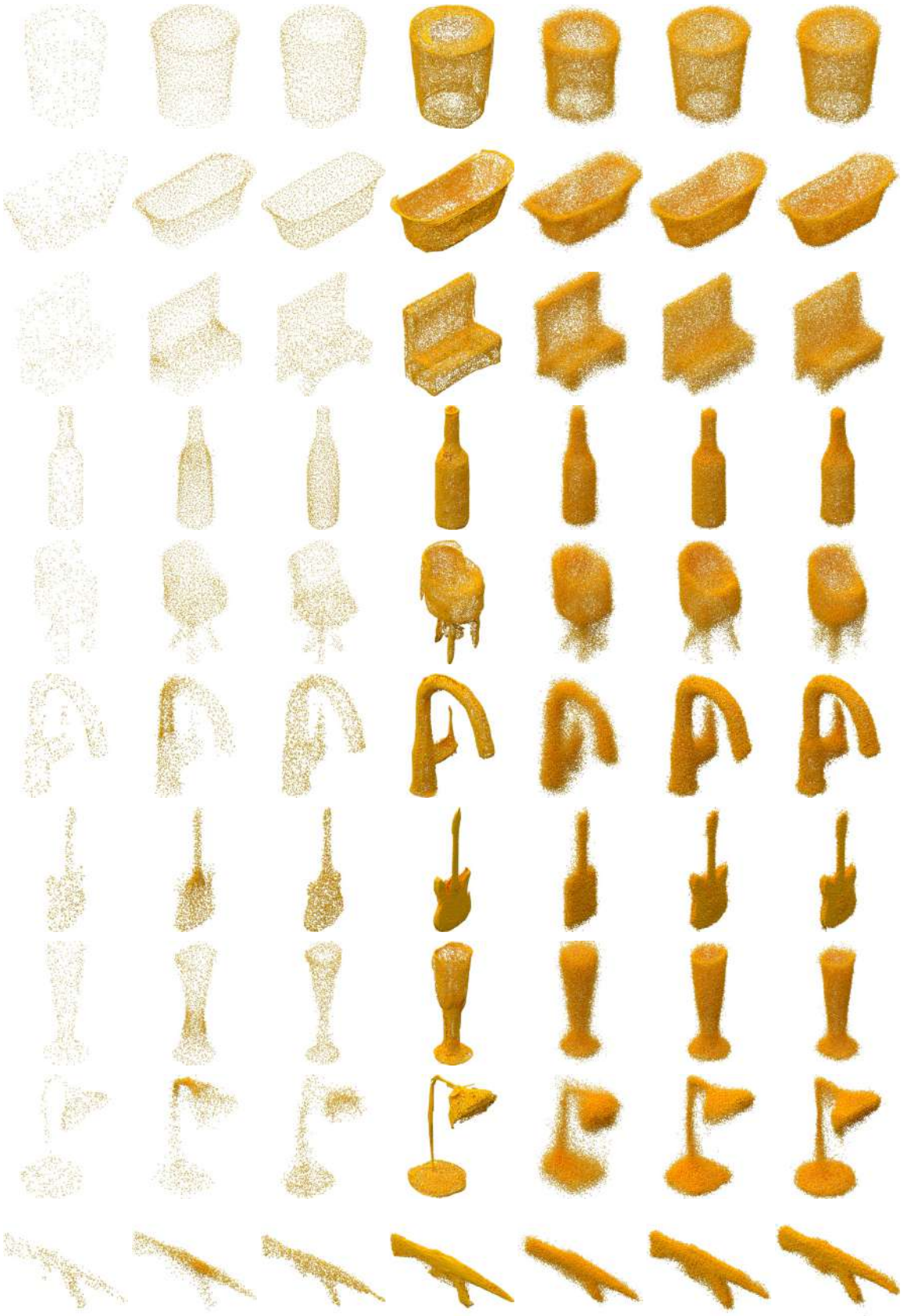

Figure 4.7: Qualitative comparison of the models from Table 4.3 for the autoencoding task with sparse (512 points) inputs.

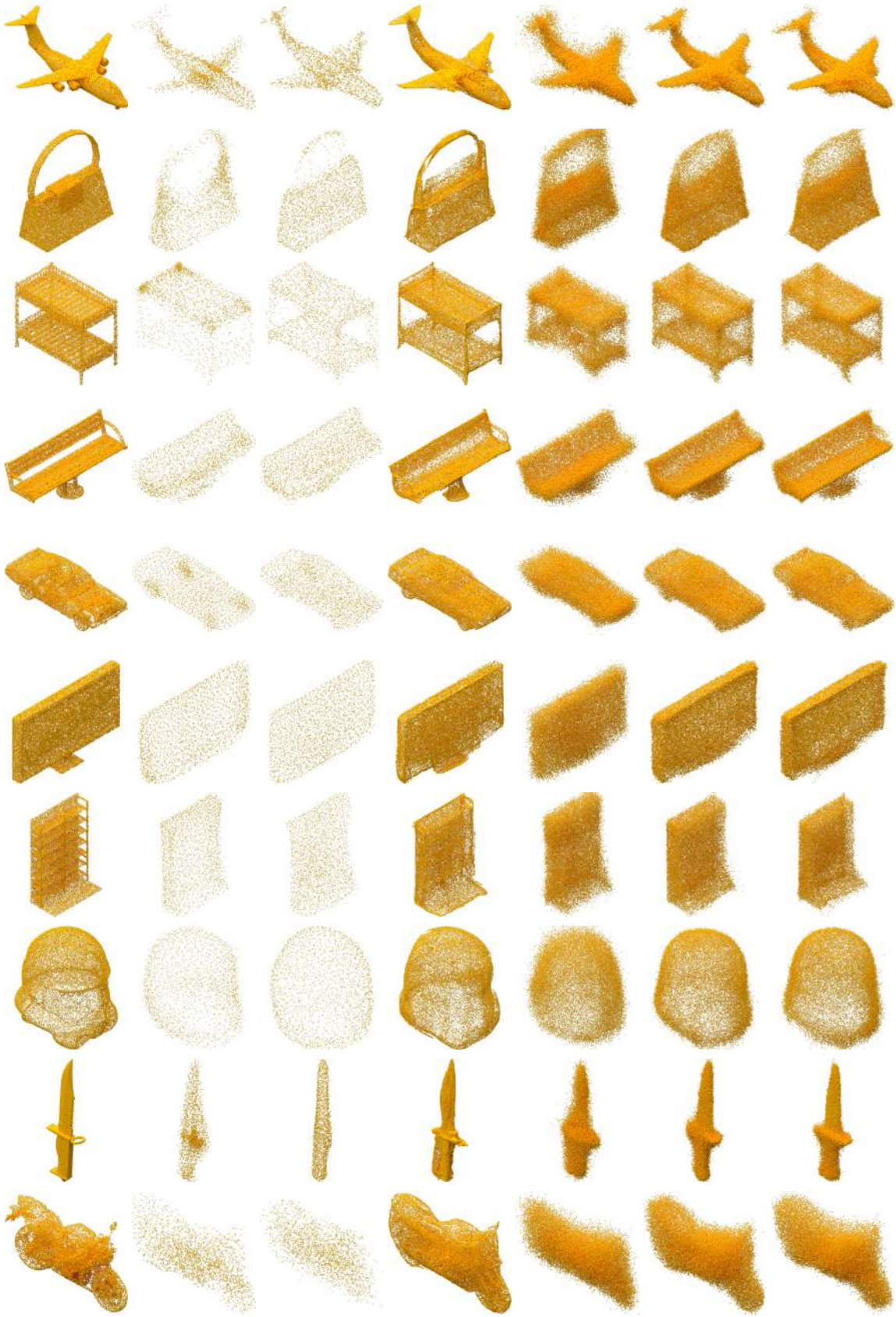

Figure 4.8: Qualitative comparison of the models from Table 4.3 for the autoencoding task with dense ($32,768$ points) inputs.

Model	CD↓, $\times 10^3$	EMD↓, $\times 10^2$	F1↑, $\tau = 0.001, \%$
PRN (Klokov et al., 2019)	7.56	11.00	**53.1**
AtlasNet (Groueix et al., 2018)	**5.34**	12.54	52.2
DCG (Wang et al., 2019a)	6.35	18.94	45.7
Pixel2Mesh† (Wang et al., 2018)	5.91	13.80	-
DPF-Nets	5.51	**10.95**	52.4
Oracle	1.10	5.70	84.0

Table 4.4: Single-view reconstruction results. †: results taken from (Wang et al., 2018).

Our DPF-Nets produce overall smoother reconstructions, but, on the other hand, suffer from more noise.

4.4.5 Single-View Reconstruction

In this section we test DPF-Nets on the inference of 3D point clouds from single images. The architecture used for this specific task is depicted in Figure 4.3 and detailed in Section 4.3.2. We compare our results to recent state-of-the-art methods in the field. This includes: the voxel-based PRN (Klokov et al., 2019), point cloud-based approaches of AtlasNet (Groueix et al., 2018) and DCG (Wang et al., 2019a), and the mesh-based Pixel2Mesh (Wang et al., 2018).

Although convenient, in general, comparison to voxel-based approaches should be taken with a grain of salt, since it is biased. To compute the proposed metrics either ground truth or reconstructed voxelized shapes are fed to the marching cubes algorithm to obtain final meshes which are used to sample point clouds. Resulting ground truth meshes in that case are crude approximations of the original meshes, used in the evaluation of the point cloud and mesh-based approaches. Moreover, there are cases both in voxelized data and reconstructions, when the marching cubes algorithm fails to output meshes. We removed such samples (approximately 0.5% of samples) from the final metric calculations for the voxel based model.

The results in Table 4.4 show that DPF-Net clearly outperforms earlier works in terms of the EMD metric. It also achieves best results in terms of the F1-score among point cloud and mesh-based models. In terms of CD, similarly to autoencoding it is outperformed only by AtlasNet with a small margin. This validates the ability of normalizing flows to capture complex distributions in 3D and to model shape surfaces.

Qualitative single-view reconstruction results can be found in Figure 4.9. Note that a single reconstruction model has been trained across all 13 classes for both AtlasNet and DPF-Net. Similarly to the autoencoding task, compared to AtlasNet our approach produces more evenly distributed point clouds without sharp dense clusters, but introduces more noise.

4.5 Conclusion

We presented DPF-Networks, a generative model for point clouds of arbitrary size. DPF-Nets are based on a latent variable model and use normalizing flows with affine coupling layers to construct a flexible, yet tractable, shape conditional density on 3D points, and an expressive latent shape space prior. They are trained akin to VAEs, using a permutation invariant point cloud encoder as approximate posterior distribution over the latent shape space.

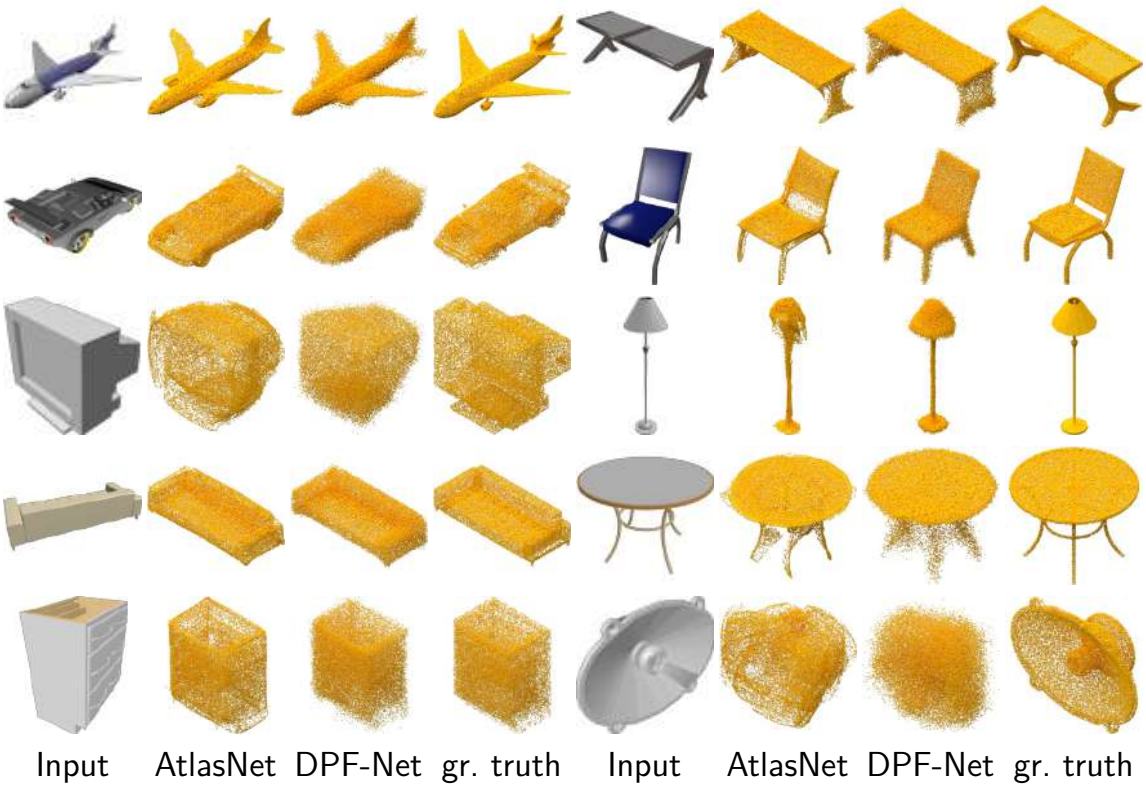

Figure 4.9: Qualitative comparison for single-view reconstruction task.

The evaluation on the ShapeNet dataset demonstrates that DPF-nets improve generative performance metrics over previous work in most metrics and classes. Compared to PointFlow which is based on continuous normalizing flows, our model is between one and two orders of magnitude faster to train and sample from. Applied to single view reconstruction, DPF-Nets outperform state-of-the-art methods, hence showing promising capabilities in 3D shape modeling.

CHAPTER 5

Conclusion

The work presented in this thesis focuses on 3D shape modeling. It proposes contributions to both pure generative and image conditioned 3D shape inference models for various 3D shape representations. To achieve this, this work combines the use of deep neural networks suitable for considered 3D shape representations and corresponding probabilistic models providing principled training objectives for the presented DNNs. As the result, two separate models were proposed: a probabilistic model, which formalizes the single-view 3D shape inference task, and another efficient probabilistic model for 3D point cloud generation, based on conditional coupling normalizing flows. We now summarize the impact of the presented works.

Probabilistic Reconstruction Networks

We propose a family of latent variable models that provides a meaningful probability-based interpretation of the single-view 3D shape inference task. It extensively describes all image conditioning options in all the model components, abstracting away from target 3D shape representation and particular implementations of image conditioning mechanisms. Moreover, different approximations of model likelihood provide a link to different model training regimes: straightforward inference from images by the means of image encoder and 3D shape decoder, or additional use of 3D shape encoding component for improved training. Such formalization allows to effectively decouple the impact of mentioned model components and design choices on the final quality of reconstruction. Proper categorisation of prior work with respect to these attributes helps to identify potential underexamined combinations and can help to focus further research efforts.

Experimental results obtained for voxelized shapes from the ShapeNet dataset show that variational training, allowed by the approximate posterior of the latent variable implemented by additional 3D shape encoder, improves the single-view 3D shape inference performance compared to training with more common objective based on sampling from latent variable prior. We also show that probabilistic objectives perform better compared to deterministic counterparts. This evidence supports the idea that it is beneficial to model conditional generation in a probabilistic manner in case when the condition defines the output very loosely (3D shape inference from a single image). Evaluation of common point cloud metrics calculated from voxel occupancy grids showed that despite the low grid resolution, our results can be competitive with point cloud based models.

Discrete Point Flow Networks

In case of point cloud modeling, probabilistic approaches have even more benefits compared to deterministic options. The occupancy grids, studied in our first contribution are regular and stable in a sense that for a given shape the fixed resolution occupancy grid is deterministic. It is not the case for point clouds, since any shape can be represented by an infinite number of point clouds sampled from the shape surface. Our probabilistic model for point clouds allows to capture this by means of learning and sampling distributions of points from shape surfaces. Thus, our approach for point cloud modeling naturally supports varying cardinality of point sets and accounts for variability in different point cloud samples representing the same 3D geometry.

Our *Discrete Point Flow Networks* are structurally similar to the prior work of (Yang et al., 2019). However, we adapt discrete invertible normalizing flows to the framework and propose a novel modification of coupling flows, which 1) includes a conditioning mechanism; 2) is designed for application to unstructured multidimensional set data, unlike original unconditional coupling flows proposed for multidimensional vector data. To further strengthen our contribution, we transformed the proposed generative model into a single-view reconstruction model using a variational approach with image-conditioned global latent variable prior, inspired from our first contribution.

In practice, the development of our conditional point cloud coupling flows allowed us to achieve significant speedups both in training and sampling without compromising for worse quality or diversity of samples, all compared to related model of (Yang et al., 2019) using continuous flows. Without direct optimization of the considered metrics our approach achieves state-of-the-art autoencoding and single-view reconstruction performance in EMD, which corresponds to sampling of more balanced point clouds distributed evenly over the target shape surfaces.

5.1 Future Work

Based on the expertise developed over the course of my work summarized in this thesis and personal interests, in the following, I describe a number of directions for future work.

Other Options for Point Cloud Modeling

PointFlow and our DPF-Net implementations of the probabilistic model for point cloud generation greatly rely on the use of normalizing flows in the conditional decoder $p_\theta(\mathbf{x}|\mathbf{z})$. According to the model, every point is conditioned only by the same latent variable sample. This makes every predicted point indistinguishable from other points in terms of correspondence to ground truth points.

Invertability of normalizing flows, used in PointFlow and our DPF-Net, allows training without correspondence between ground truth and sampled point clouds. These models do not sample points during training. Instead, inverse flow calculations transform ground truth surface points into samples from a flow base Gaussian distribution. As the result, the complex likelihood of ground truth point clouds is expressed through the trivially calculable likelihood of the said samples from the Gaussian, as discussed in Chapter 4. Such a training method could not be possible without the invertability of the flow decoder.

In order to fully utilize all the developed advances in DNN architectures for point cloud representation, other less restrictive probabilistic models could be developed that do not require invertability from its decoding components.

One of the most straightforward extensions of the considered model could be obtained by introduction of additional set of local per point latent variables $\{\mathbf{l}\}_{i=1}^{N}$:

$$p(X) = \int \prod_{i=1}^{N} p(\mathbf{x}_i|\mathbf{l}_i, \mathbf{z}) p(\mathbf{l}_i|\mathbf{z}) p(\mathbf{z}) \mathrm{d}\mathbf{l}_i \mathrm{d}\mathbf{z}. \tag{5.1}$$

In such a model, a variational approximation of the likelihood is possible with approximate posteriors for both global $q(\mathbf{z}|X)$ and local $q(\mathbf{l}_i|X, \mathbf{z})$ per-point distributions. If these local posteriors are conditioned on individual ground truth points $\mathbf{x}_i$ during training, this will automatically provide correspondence between pairs of sampled and ground truth points. This allows to implement point decoder by generic DNN architectures for point clouds, developed in numerous related approaches.

Unfortunately, this model reintroduces the same correspondence issue for local latent variable priors and posteriors, unless all the local priors are the same distribution (all-to-one correspondence is automatic in that case). Our initial experiments with simple united local latent variable prior showed that such a model is not effective in a generative setting. Other, more expressive types of local priors could be considered to solve this issue, otherwise the probabilistic model should be modified further to completely exclude this problem in the first place.

Besides modeling with only the invertible decoding components, there are other limitations of considered probabilistic models, that could be lifted. The models presented in this work all rely on the ability of a sole latent global variable to capture not only the variation in shape topologies but all the intricacies of 3D shape geometric detail. Hierarchical models similar to the one stated earlier could help to relax this requirement from the global variable and have recently shown success for image generation task (Vahdat and Kautz, 2020). It could be a challenging and rewarding task to establish effective hierarchical models for various 3D data representations, including point clouds.

Probabilistic Models for Mesh Representations

The mesh representation of 3D shapes remains to be the most challenging among explicit 3D geometry representations for probabilistic modeling. At the same time, it is often a desired output of a 3D shape reconstruction model, since it is explicit, expressive, fully represents shape geometry (unlike *e.g.* point clouds), and could be used in numerous further applications. The main difficulty with it lies in the requirement to model not only the positions of the vertices, that is similar to point cloud modeling, but also to model connectivity between the vertices. In the worst case, with brute pairwise consideration of all the vertices, connectivity modeling leads to quadratic complexity over the number of vertices.

The main reason to consider probabilistic models for mesh representation is their potential to overcome a common problem with existing mesh-based solutions for generative and inference tasks: their inability to capture various topologies. This problem stems from the deformation-based framework that is often used with mesh representation, *e.g.* (Tan et al., 2018; Wang et al., 2018; Gao et al., 2019). In this framework, an initial fixed topology mesh prototype or template is locally deformed to fit its surface geometry to the ground truth without any

changes in the initial topology. Explicit modeling of mesh connectivity with probabilistic approaches is capable of overcoming these topological restrictions in mesh reconstructions.

To our knowledge, only the work of (Nash et al., 2020) presents an autoregressive probabilistic model for meshes that model connectivity explicitly. Although it solves the task, the proposed model is based on autoregressive transformer networks which usually are demanding both in terms of memory and training/inference times. The recent approach of (Chen et al., 2020) is also capable of producing meshes as collections of convex primitives, however it uses a deterministic model and considers only inference tasks. Probabilistic reformulation of such an approach could extend this model to generative applications, introduce accountability for variation in reconstructed 3D shapes and may bring further improvements in the reconstruction quality.

Going beyond Pure Geometry Learning

As described in Chapter 1, the geometric aspect of 3D data does not fully describe it. Besides the geometry, there are color, reflectance and, possibly, extended shading information, which is required for proper visual representation of 3D data. The use of all the modalities of 3D data for realistic visualization was and still it the main topic of computer graphics. However, principled image rendering techniques, creating the best results are computationally demanding and may require tremendous amounts of resources. Recently, the success in the data-driven super-resolution and reflection rendering tasks have already found use in computer graphics applications and solutions, such as NVIDIA's deep learning super sampling and ray tracing technologies used for computer game graphics.

Joint data-driven modeling of all the modalities of 3D data is a challenging task, that can enable multiple generative and inference applications. Success has already been shown for normal inference from point coordinates (Guerrero et al., 2018), which may improve the surface reconstruction and enhance realistic shading during rendering for ill-defined 3D shapes (such as point clouds). The work of (Mildenhall et al., 2020) on the data-driven novel view synthesis have potential to provide an effective and efficient alternative to standard rendering for image editing, animation and robotic spacial understanding end applications. Similarly to purely geometric approaches, advances in joint 3D data modeling could enable various content creation applications: generation or inference from partial inputs of complete 3D shapes and scenes including geometry, appearance, and shading; geometric and appearance enhancement of 3D data, similar to super resolution; realistic inference-based insertion of 3D shapes into real images with correct shading and reflections (without the use of explicit rendering techniques) and other image modification tasks.

Bibliography

Achlioptas, P., Diamanti, O., Mitliagkas, I., and Guibas, L. (2018). Learning representations and generative models for 3D point clouds. In *ICML*.

Bau, D., Zhu, J.-Y., Strobelt, H., Lapedriza, A., Zhou, B., and Torralba, A. (2020). Understanding the role of individual units in a deep neural network. *Proceedings of the National Academy of Sciences*, 117(48):30071–30078.

Behrmann, J., Grathwohl, W., Chen, R., Duvenaud, D., and Jacobsen, J.-H. (2019). Invertible residual networks. In *ICML*.

Bhattacharyya, A., Hanselmann, M., Fritz, M., Schiele, B., and Straehle, C.-N. (2019). Conditional flow variational autoencoders for structured sequence prediction. In *NeurIPS Workshop on Machine Learning for Autonomous Driving*.

Bishop, C. (2006). *Pattern recognition and machine learning*. Spinger-Verlag.

Bogo, F., Romero, J., Loper, M., and Black, M. J. (2014). FAUST: Dataset and evaluation for 3D mesh registration. In *CVPR*.

Boscaini, D., Masci, J., Rodolà, E., and Bronstein, M. (2016). Learning shape correspondence with anisotropic convolutional neural networks. In *NeurIPS*.

Brock, A., Lim, T., Ritchie, J., and Weston, N. (2016). Generative and discriminative voxel modeling with convolutional neural networks. In *NeurIPS 3D deep learning workshop*.

Bronstein, M., Bruna, J., Szlam, A., LeCun, Y., and Vandergyst, P. (2017). Geometric deep learning: going beyond Euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. (2013). Spectral networks and locally connected networks on graphs. *arXiv preprint*.

Cai, R., Yang, G., Averbuch-Elor, H., Hao, Z., Belongie, S., Snavely, N., and Hariharan, B. (2020). Learning gradient fields for shape generation. In *ECCV*.

Chang, A., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., and Yu, F. (2015). ShapeNet: An information-rich 3D model repository. *arXiv preprint*.

Chen, T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. (2018). Neural ordinary differential equations. In *NeurIPS*.

Chen, X., Kingma, D., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., Sutskever, I., and Abbeel, P. (2017). Variational lossy autoencoder. In *ICLR*.

Chen, Z., Tagliasacchi, A., and Zhang, H. (2020). Bsp-net: Generating compact meshes via binary space partitioning. In *CVPR*.

Chen, Z. and Zhang, H. (2019). Learning implicit fields for generative shape modeling. In *CVPR*.

Choy, C., Xu, D., Gwak, J.-Y., Chen, K., and Savarese, S. (2016). 3D-R2N2: A unified approach for single and multi-view 3d object reconstruction. In *ECCV*.

Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T., and Nießner, M. (2017). Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*.

Deco, G. and Brauer, W. (1995). Higher order statistical decorrelation without information loss. In *NeurIPS*.

Defferrard, M., Bresson, X., and Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS*.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2017). Density estimation using Real NVP. In *ICLR*.

Erler, P., Guerrero, P., Ohrhallinger, S., Mitra, N. J., and Wimmer, M. (2020). Points2surf learning implicit surfaces from point clouds. In *ECCV*.

Fan, H., Su, H., and Guibas, L. (2017). A point set generation network for 3D object reconstruction from a single image. In *CVPR*.

Finetti, B. D. (1937). La prévision : ses lois logiques, ses sources subjectives. In *Annales de l'institut Henri Poincaré*.

Fu, H., Jia, R., Gao, L., Gong, M., Zhao, B., Maybank, S., and Tao, D. (2020). 3d-future: 3d furniture shape with texture. *arXiv preprint*.

Gabeur, V., Franco, J.-S., Martin, X., Schmid, C., and Rogez, G. (2019). Moulding humans: Non-parametric 3d human shape estimation from single images. In *ICCV*.

Gao, L., Lai, Y.-K., Liang, D., Chen, S.-Y., and Xia, S. (2016). Efficient and flexible deformation representation for data-driven surface modeling. *ACM Transactions on Graphics*.

Gao, L., Yang, J., Wu, T., Yuan, Y.-J., Fu, H., Lai, Y.-K., and Zhang, H. (2019). Sdm-net: Deep generative network for structured deformable mesh. *ACM Transactions on Graphics*.

Girdhar, R., Fouhey, D., Rodriguez, M., and Gupta, A. (2016). Learning a predictable and generative vector representation for objects. In *ECCV*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *NeurIPS*.

Gordon, J., Bronskill, J., Bauer, M., Nowozin, S., and Turner, R. (2019). Meta-learning probabilistic inference for prediction. In *ICLR*.

Graham, B., Engelcke, M., and van der Maaten, L. (2018). 3D semantic segmentation with submanifold sparse convolutional networks. In *CVPR*.

Grathwohl, W., Chen, R., Bettencourt, J., Sutskever, I., and Duvenaud, D. (2019). FFJORD: Free-form continuous dynamics for scalable reversible generative models. In *ICLR*.

Groueix, T., Fisher, M., Kim, V., Russell, B., and Aubry, M. (2018). A papier-mâché approach to learning 3D surface generation. In *CVPR*.

Guerrero, P., Kleiman, Y., Ovsjanikov, M., and Mitra, N. J. (2018). Pcpnet learning local shape properties from raw point clouds. In *Computer Graphics Forum*.

Han, X., Li, Z., Huang, H., Kalogerakis, E., and Yu, Y. (2017). High-resolution shape completion using deep neural networks for global structure and local geometry inference. In *ICCV*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep residual learning for image recognition. In *CVPR*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Identity mappings in deep residual networks. In *ECCV*.

Henaff, M., Bruna, J., and LeCun, Y. (2015). Deep convolutional networks on graph-structured data. *arXiv preprint*.

Henderson, P. and Ferrari, V. (2018). Learning to generate and reconstruct 3d meshes with only 2d supervision. In *BMVC*.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*.

Hu, Z., Yang, Z., Salakhutdinov, R., and Xing, E. P. (2018). On unifying deep generative models. In *ICLR*.

Insafutdinov, E. and Dosovitskiy, A. (2018). Unsupervised learning of shape and pose with differentiable point clouds. In *NeurIPS*.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2018). Progressive growing of GANSs for improved quality, stability, and variation. In *ICLR*.

Kazmi, I. K., You, L., and Zhang, J. (2013). A survey of 2d and 3d shape descriptors. *International Conference Computer Graphics, Imaging and Visualization*.

Kingma, D. and Dhariwal, P. (2018). Glow: Generative flow with invertible 1×1 convolutions. In *NeurIPS*.

Kingma, D., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. (2016). Improved variational inference with inverse autoregressive flow. In *NeurIPS*.

Kingma, D. and Welling, M. (2014). Auto-encoding variational Bayes. In *ICLR*.

Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint*.

Klokov, R., Boyer, E., and Verbeek, J. (2020). Discrete point flow networks for efficient point cloud generation. In *ECCV*.

Klokov, R. and Lempitsky, V. (2017). Escape from cells: Deep Kd-networks for the recognition of 3D point cloud models. In *ICCV*.

Klokov, R., Verbeek, J., and Boyer, E. (2019). Probabilistic reconstruction networks for 3D shape inference from a single image. In *BMVC*.

Knapitsch, A., Park, J., Zhou, Q., and Koltun, V. (2017). Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4).

Kobyzev, I., Prince, S., and Brubaker, M. (2019). Normalizing flows: An introduction and review of current methods. *arXiv preprint*.

Koch, S., Matveev, A., Jiang, Z., Williams, F., Artemov, A., Burnaev, E., Alexa, M., Zorin, D., and Panozzo, D. (2019). Abc: A big cad model dataset for geometric deep learning. In *CVPR*.

Krizhevsky, A., Sutskever, I., and Hinton, G. (2012). ImageNet classification with deep convolutional neural networks. In *NeurIPS*.

Kurenkov, A., Ji, J., Garg, A., Mehta, V., Gwak, J.-Y., Choy, C., and Savarese, S. (2018). Deformnet: Free-form deformation network for 3d shape reconstruction from a single image. In *WACV*.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lewiner, T., Lopes, H., Vieira, A., and Tavares, G. (2003). Efficient implementation of marching cubes' cases with topological guarantees. *J. Graphics, GPU, & Game Tools*, 8(2):1–15.

Li, C.-L., Zaheer, M., Zhang, Y., Poczos, B., and Salakhutdinov, R. (2018a). Point cloud GAN. *arXiv preprint*.

Li, Y., Bu, R., Sun, M., Wu, W., Di, X., and Chen, B. (2018b). Pointcnn: Convolution on x-transformed points. In *Advances in neural information processing systems*.

Lin, T., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *ECCV*.

Liu, Z., Tang, H., Lin, Y., and Han, S. (2019). Point-voxel cnn for efficient 3d deep learning. In *NeurIPS*.

Lopez-Paz, D. and Oquab, M. (2017). Revisiting classifier two-sample tests. In *ICLR*.

Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization. In *ICLR*.

Lu, Y. and Huang, B. (2020). Structured output learning with conditional generative flows. In *AAAI*.

Lucas, T., Shmelkov, K., Alahari, K., Schmid, C., and Verbeek, J. (2019). Adaptive density estimation for generative models. In *NeurIPS*.

Mandikal, P., Navaneet, K., Agarwal, M., and Babu, R. (2018). 3D-LMNet: Latent embedding matching for accurate and diverse 3D point cloud reconstruction from a single image. In *BMVC*.

Masci, J., Boscaini, D., Bronstein, M. M., and Vandergheynst, P. (2015). Geodesic convolutional neural networks on riemannian manifolds. In *ICCV Workshops*.

Maturana, D. and Scherer, S. (2015). VoxNet: A 3D convolutional neural network for real-time object recognition. In *IROS*.

Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., and Geiger, A. (2019). Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*.

Michalkiewicz, M., Pontes, J., Jack, D., Baktashmotlagh, M., and Eriksson, A. (2019). Implicit surface representations as layers in neural networks. In *ICCV*.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2020). Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*.

Monti, F., Boscaini, D., Masci, J., Rodolà, E., Svoboda, J., and Bronstein, M. (2017). Geometric deep learning on graphs and manifolds using mixture model CNNs. In *CVPR*.

Nash, C., Ganin, Y., Eslami, S. M. A., and Battaglia, P. W. (2020). Polygen: An autoregressive generative model of 3d meshes. In *ICML*.

Ng, J. Y.-H., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., and Toderici, G. (2015). Beyond short snippets: Deep networks for video classification. In *CVPR*.

Papamakarios, G., Nalisnick, E., Rezende, D., Mohamed, S., and Lakshminarayanan, B. (2019). Normalizing flows for probabilistic modeling and inference. *arXiv preprint*.

Park, J., Florence, P., Straub, J., Newcombe, R., and Lovegrove, S. (2019). DeepSDF: Learning continuous signed distance functions for shape representation. In *CVPR*.

Perez, E., Strub, F., Vries, H. D., Dumoulin, V., and Courville, A. (2018). FiLM: Visual reasoning with a general conditioning layer. In *AAAI*.

Pumarola, A., Popov, S., Moreno-Noguer, F., and Ferrari, V. (2020). C-Flow: Conditional generative flow models for images and 3D point clouds. In *CVPR*.

Qi, C., Su, H., Mo, K., and Guibas, L. (2017a). Pointnet: Deep learning on point sets for 3D classification and segmentation. In *CVPR*.

Qi, C., Yi, L., Su, H., and Guibas, L. (2017b). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*.

Reddi, S., Kale, S., and Kumar, S. (2018). On the convergence of Adam and beyond. In *ICLR*.

Reizenstein, J., Shapovalov, R., Henzler, P., Sbordone, L., Labatut, P., and Novotny, D. (2021). Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *International Conference on Computer Vision*.

Rezende, D. and Mohamed, S. (2015). Variational inference with normalizing flows. In *ICML*.

Rezende, D., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *ICML*.

Richter, S. R. and Roth, S. (2018). Matryoshka Networks: Predicting 3d geometry via nested shape layers. In *CVPR*.

Riegler, G., Ulusoy, A., and Geiger, A. (2017). OctNet: Learning deep 3D representations at high resolutions. In *CVPR*.

Shin, D., Fowlkes, C. C., and Hoiem, D. (2018). Pixels, voxels, and views: A study of shape representations for single view 3d object shape prediction. In *CVPR*.

Sinha, A., Bai, J., and Ramani, K. (2016). Deep learning 3D shape surfaces using geometry images. In *ECCV*.

Smith, E. and Meger, D. (2017). Improved adversarial systems for 3d object generation and reconstruction. In *CoRL*.

Soltani, A., Huang, H., Wu, J., Kulkarni, T., and Tenenbaum, J. (2017). Synthesizing 3D shapes via modeling multi-view depth maps and silhouettes with deep generative networks. In *CVPR*.

Su, H., Fan, H., and Guibas, L. (2017). A point set generation network for 3D object reconstruction from a single image. In *CVPR*.

Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M.-H., and Kautz, J. (2018). SPLATNet: Sparse lattice networks for point cloud processing. In *CVPR*.

Su, H., Maji, S., Kalogerakis, E., and Learned-Miller, E. (2015). Multi-view convolutional neural networks for 3D shape recognition. In *ICCV*.

Sun, X., Wu, J., Zhang, X., Zhang, Z., Zhang, C., Xue, T., Tenenbaum, J., and Freeman, W. (2018). Pix3D: Dataset and methods for single-image 3D shape modeling. In *CVPR*.

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*.

Tan, Q., Gao, L., Lai, Y.-K., and Xia, S. (2018). Variational autoencoders for deforming 3d mesh models. In *CVPR*.

Tatarchenko, M., Dosovitskiy, A., and Brox, T. (2017). Octree generating networks: Efficient convolutional architectures for high-resolution 3D outputs. In *ICCV*.

Tatarchenko, M., Richter, S. R., Ranftl, R., Li, Z., Koltun, V., and Brox, T. (2019). What do single-view 3d reconstruction networks learn? In *CVPR*.

Tulsiani, S., Zhou, T., Efros, A., and Malik, J. (2017). Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *CVPR*.

Vahdat, A. and Kautz, J. (2020). NVAE: A deep hierarchical variational autoencoder. In *NeurIPS*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *NeurIPS*.

Verma, N., Boyer, E., and Verbeek, J. (2018). Feastnet: Feature-steered graph convolutions for 3D shape analysis. In *CVPR*.

Vinyals, O., Fortunato, M., and Jaitly, N. (2015). Pointer networks. In *NeurIPS*.

Wang, H., Jiang, Z., Yi, L., Mo, K., Su, H., and Guibas, L. J. (2020). Rethinking sampling in 3d point cloud generative adversarial networks. *arXiv preprint*.

Wang, K., Chen, K., and Jia, K. (2019a). Deep cascade generation on point sets. In *IJCAI*.

Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., and Jiang, Y. (2018). Pixel2Mesh: Generating 3D mesh models from single RGB images. In *ECCV*.

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. (2019b). Dynamic graph cnn for learning on point clouds. *ACM Transactions On Graphics*.

Wiles, O. and Zisserman, A. (2017). SilNet: Single- and multi-view reconstruction by learning from silhouettes. In *BMVC*.

Wu, J., Wang, Y., Xue, T., Sun, X., Freeman, B., and Tenenbaum, J. (2017). MarrNet: 3D shape reconstruction via 2.5D sketches. In *NeurIPS*.

Wu, J., Zhang, C., Xue, T., Freeman, W., and Tenenbaum, J. (2016). Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In *NeurIPS*.

Wu, J., Zhang, C., Zhang, X., Zhang, Z., Freeman, W., and Tenenbaum, J. (2018). Learning shape priors for single-view 3D completion and reconstruction. In *ECCV*.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. (2015). 3D ShapeNets: A deep representation for volumetric shapes. In *CVPR*.

Yan, X., Yang, J., Yumer, E., Guo, Y., and Lee, H. (2016). Perspective Transformer Nets: Learning single-view 3D object reconstruction without 3D supervision. In *NeurIPS*.

Yang, G., Huang, X., Hao, Z., Liu, M.-Y., Belongie, S., and Hariharan, B. (2019). PointFlow: 3D point cloud generation with continuous normalizing flows. In *ICCV*.

Yi, L., Shao, L., Savva, M., Huang, H., Zhou, Y., Wang, Q., Graham, B., Engelcke, M., Klokov, R., Lempitsky, V., et al. (2017). Large-scale 3d shape reconstruction and segmentation from shapenet core55. *arXiv preprint*.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R., and Smola, A. (2017). Deep sets. In *NeurIPS*.

Zhang, Z. (2012). Microsoft kinect sensor and its effect. *IEEE multimedia*.

Zhou, L., Du, Y., and Wu, J. (2021). 3d shape generation and completion through point-voxel diffusion. In *ICCV*.

Zhou, Q. and Jacobson, A. (2016). Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint*.