3D Representation Learning Endowed by Optimal Transport

Thesis submitted in partial fulfillment
of the requirements for the degree of

*Master of Science in **Computer Science and Engineering** by Research*

by

Siddharth Katageri
2021701018
siddharth.katageri@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
March 2024

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled **"3D Representation Learning Endowed by Optimal Transport"** by Siddharth Katageri, has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Adviser: Dr. Charu Sharma

To by beloved mummi, appa, and sister

Acknowledgments

It was indeed a transforming journey here at IIIT Hyderabad. A journey that has been both challenging and rewarding. It has taught me how to think, learn, and communicate. I would like to express my deepest gratitude to all those who have contributed to the completion of my thesis.

I would like to sincerely thank my advisor Prof. Charu Sharma for her unwavering support, insightful feedback, and endless patience throughout this entire journey. Her expertise and guidance have been instrumental in shaping the direction of my research and refining my academic skills. She always used to be my strong pillar of support whenever I used to feel low or when times were getting hard. She always used to have my back. I would also like to thank Prof. Uma Mudenagudi for igniting the spark of research in me. Her guidance, support, and motivation have been invaluable to me.

I would like to thank my friends and seniors - Rudrabha, Siddhanth, Seshadri, Aditya Arun, Shubham, Madhav, Zeeshan, Varun, Shivanshu, George, Darshana, Vivek, Joy, Alik, Aditya, Bipasha, Shanthika, Abhinaba, Chandradeep, Ravi, Pranav for making the lab a fun place to stay. Huge thanks to Soumya and Darshan for supporting, guiding, encouraging, and always being there with me. Also, huge thanks to Aditya, Chetan, and Arpit, my badminton team, with whom I regularly played badminton and had so much fun together. I would like to thank and appreciate my batch friends Dhruv, Amruth, Prateek, Bhoomendra, Nayan, Madan, Shishir, and others who have been the best sport for the entire time at IIIT.

Lastly, I would like to thank my family and friends outside IIIT. I am thankful to my family for their unwavering support and encouragement. Thank you, mummi and appa for always believing in my abilities and constantly wishing for my well-being. Your constant motivation and understanding have helped a lot in the demanding periods of the journey. Also, thank you to my sister for being a sensible kid.

Abstract

Consider an autonomous agent capable of obeying the following instruction "Go and clean the coffee spilled on the dining table". To perfectly execute this instruction, the agent needs to have a precise understanding of its dynamic 3D environment. The final task can be broken down into the following sub-tasks; 3D grounding, 3D semantic understanding, and 3D motion planning. To excel in all these tasks, 3D representation learning plays an important role. Motivated to contribute towards creating systems that can perceive and act in real 3D words, in this thesis, we propose two novel methods for 3D representation learning.

The first portion of this thesis focuses on the tasks of unsupervised domain adaptation (UDA) for 3D point clouds. The point cloud data acquisition procedures manifest themselves as significant domain discrepancies and geometric variations among both similar and dissimilar classes. The standard domain adaptation methods developed for images do not directly translate to point cloud data because of their complex geometric nature. Existing works mainly focused on designing a self-supervised task to improve adaptation performance. We propose a new UDA architecture for point cloud classification that benefits from multimodal contrastive learning to get better class separation in both domains individually. Further, the use of optimal transport aims at learning source and target data distributions jointly to reduce the cross-domain shift and provide a better alignment. We conduct a comprehensive empirical study on PointDA 10 and GraspNetPC-10 and show that our method achieves state-of-the-art performance on GraspNetPC-10 (with $\approx$ 4-12% margin) and best average performance on PointDA-10. Our ablation studies and decision boundary analysis also validate the significance of our contrastive learning module and OT alignment.

In the second portion of the thesis, we explore learning Wasserstein embeddings for point clouds towards multiple downstream tasks. As learning embeddings of any data largely depends on the ability of the target space, we propose to embed point clouds as discrete probability distributions in Wasserstein space. We build a contrastive learning setup to learn Wasserstein embeddings that can be used as a pre-training method with or without supervision towards any downstream task. We show that the features captured by Wasserstein embeddings are better in preserving the point cloud geometry, including both global and local information, thus resulting in improved quality embeddings. We perform exhaustive experiments and demonstrate the effectiveness of our method for point cloud classification, transfer learning, segmentation, and interpolation tasks over multiple datasets, including synthetic and real-world objects. We also compare against recent methods that use Wasserstein space and show that

our method outperforms them in all downstream tasks. Additionally, our study reveals a promising interpretation of capturing critical points of point clouds that makes our proposed method self-explainable.

We hope this work motivates future research in utilizing optimal transport for understanding the real 3D world. We also hope that the self-supervised approaches proposed in this thesis will act as a step towards in this direction.

Contents

List of Figures

List of Tables

Chapter 1

Introduction to 3D Representation Learning

Over the past few decades, representation learning has undergone a profound evolution, revolutionizing the way machines interpret and process data. Initially dominated by handcrafted features and shallow learning techniques, the field has witnessed a transformative shift with the advent of deep learning. This paradigm change empowers algorithms to automatically extract hierarchical and abstract features from raw data, offering unprecedented capabilities in tasks such as image recognition, natural language understanding, and audio recognition. Despite the remarkable progress, existing approaches to representation learning face inherent limitations, particularly when confronted with the intricacies of three-dimensional (3D) data. Traditional methods, designed for two-dimensional data, struggle to capture the richness and complexity inherent in 3D structures. As technology continues to leverage 3D representations in applications ranging from virtual reality to medical imaging, the need for innovative solutions to address these limitations becomes increasingly evident.

The importance of 3D representation learning extends far beyond theoretical curiosity, finding practical applications in various domains. In computer vision, where interpreting visual scenes is fundamental, the ability to accurately represent 3D structures is a critical determinant of success. Robotics [22] heavily relies on spatial awareness for navigation and object manipulation, tasks that necessitate a robust understanding of three-dimensional environments. Animation [26] is another area that has unprecedented advantages if driven by automated systems that understand motion dynamics rather than manual labor. Applications like shape synthesis and modeling [43], autonomous driving [21], and indoor navigation [45] also need an understanding of 3D scenes. Furthermore, virtual reality (VR) and augmented reality (AR) applications, designed to immerse users in lifelike experiences, demand advanced 3D representation capabilities. Consequently, the development of effective and versatile 3D representation learning methodologies is pivotal for unlocking the full potential of these technologies.

Depending on the application, there are many popular and preferred ways to represent 3D data; some of them are shown in Figure 1.1 and are introduced below

1. **Voxel grid**: This 3D representation is the direct extension of images, where the 2D image grid is extended in another orthogonal direction, making it a 3D volumetric grid. In most cases, it is not efficient for 3D deep learning as it represents both occupied and non-occupied parts of the

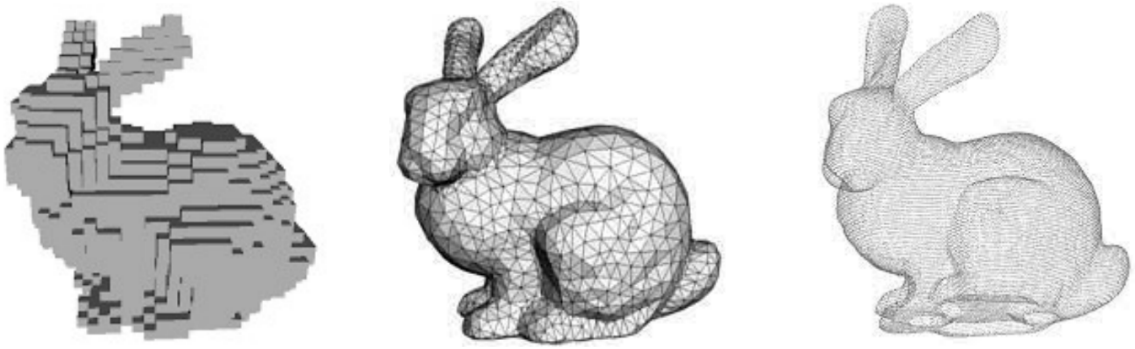


Figure 1.1: Popular types of representations for 3D data, them being, Voxel grid, Polygonal mesh, and Point cloud (from left to right).

scene, which establishes an enormous amount of unnecessary memory storage. Earlier methods processed 3D data by converting them into regular structures like voxel grids [22, 41] to employ well-explored powerful convolutional techniques. However, the transformations to voxel grids either incur loss of information or require high memory and computational complexity. The level of surface detail highly depends on the resolution of the volumetric grid. Retaining high geometry details demands high voxel resolution that increases memory usage; on the other hand, reducing the resolution impacts in loss of geometric details.

2. **Polygonal mesh**: It consists of polygons called faces that are described with a set of vertices and edges that define the surface of any 3D scene. On one hand, these edges give extra information by providing local connectivity but also make it challenging to check the properties of a mesh (*e.g.* manifoldness) for ease of running any algorithms on them. MeshCNN [16] proposes a carefully designed method to perform convolution over mesh edges, however, it only works on meshes having a fixed resolution. DiffusionNet [33] introduces a diffusion layer that is highly effective for spatial communication making the networks automatically robust to changes in mesh resolution or remeshing.

3. **Point cloud**: Unlike a polygonal mesh, point cloud do not contain any connectivity information. It only consists of a set of points in 3D space that describe the surface of any 3D scene. Methods have been developed to learn representations by directly using raw point clouds [28, 29, 38]. [28] introduces the first method to learn directly on raw point clouds, where each point from a point cloud is passed separately through a permutation invariant function to extract its point features. [29] builds upon the previous work by incorporating neighborhood information and extracting hierarchical features. Further, [38] improves on these works by extracting point features using local graphs. Advancements in scanning technologies have made getting 3D point clouds of any scene very simple. Though capturing scenes has become easy, there are various challenges to work with point cloud data. These challenges are discussed in Section 1.2.

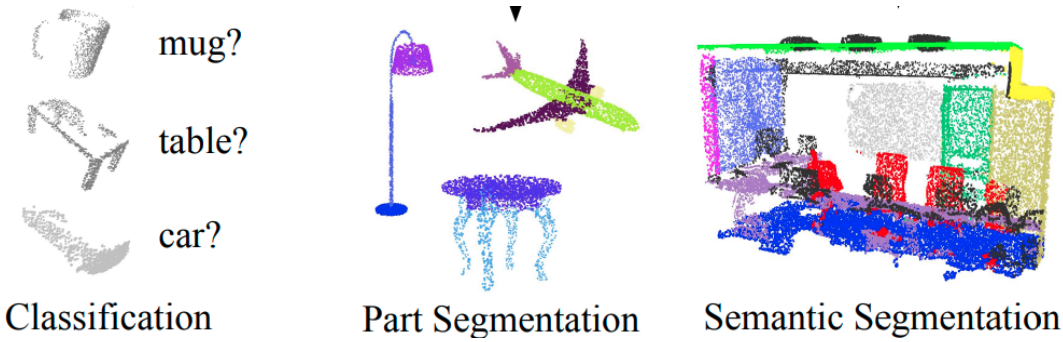


Figure 1.2: Common 3D recognition tasks performed for 3D understanding demonstrated in [28].

1.1 Common tasks performed towards 3D point cloud understanding

In order to understand a given point cloud, some common tasks are listed below and are shown in Figure 1.2. [28] proposes the first method to directly take raw point clouds as input and perform these 3D recognition tasks.

1. **3D classification**: The most basic task that one can think of for understanding a given point cloud is *identifying* what that point cloud is! More formally, predicting a class label for a given point cloud is called as 3D classification. The prediction can be based on the global geometric appearance or even based on features aggregated from local appearances.

2. **3D part segmentation**: Going a step forward for understanding a point cloud, in 3D part segmentation, we aim to predict labels for each point in the point cloud. These point labels determine which part the point belongs to. To determine different parts in the point cloud, both global and local understanding is required.

3. **3D semantic segmentation**: Along the same lines, in 3D semantic segmentation, the predicted labels for each point belong to some semantic category in the 3D scene. The 3D scene can either be an indoor or outdoor scene, having pre-defined semantic categories. Recently, methods have been developed to handle open-vocabulary semantic categories as well.

1.2 What are the challenges when working with point clouds?

In this thesis, we focus on point cloud representation for 3D data. Point clouds do not require any extra measures for processing; for example, unlike meshes, there is no traversal needed, as all points are independent. However, with its simplicity, it also poses various challenges to deal with for understanding 3D point clouds.

1. **Handling noise**: Point clouds scanned with iPhone or costly LiDAR sensors often contain lots of erroneous points. This noise arises due to inaccuracy in the sensory device or even due to environmental factors. Handling noise becomes essential for algorithms to run as expected.

2. **Irregular density**: Surface density variation is another crucial problem to handle, as the algorithm might only give importance to regions with high point density and neglect other areas.

3. **Missing regions**: Unscanned regions of the scene, called and *holes*, also pose serious challenge. Limited sensor visibility or occlusions can be the reasons for incomplete data. The algorithm should be able to reason from the surroundings and be able to handle these holes.

4. **Unstructured**: Unlike 2D images that have a fixed grid, point clouds do not have any fixed grid. Hence, widely used convolution networks cannot be directly used here. Thus, point clouds need different deep-learning algorithms to be designed for them.

5. **Unordered**: The un-orderness mainly causes a problem for deep-learning based algorithms. As all the point order permutations represent the same scene/shape, the network should map all these point order permutations to a single point. Thus, the network should be carefully designed to obey this property.

Along with these primary challenges, another very crucial challenge to handle is *domain adaptation* (DA). We discuss about this next.

1.3 Why do we need domain adaptation for point clouds?

Let's say a network is trained on a dataset for the task of 3D point cloud classification and is then evaluated on the test set of the same dataset. In this case, the network performs exceedingly well as the distribution of the test set is the same as that of the train set. However, to test the limits of the network, if we evaluate its performance on the test set of some other dataset (having the same classes), we observe a drop in the performance. Even if the classes are the same, even when both datasets (or domains) are synthetically produced, why does this happen? This is in part due to the significant differences in underlying structures (i.e., different backgrounds, orientations, illuminations, *etc*. obtained from a variety of data acquisition methods and devices), which in turn manifest themselves as geometric variations and discrepancies between the source and target point cloud domains.

Also, an important aspect of achieving cross-domain adaptation is to leverage the trained model on simulated data (easy-to-get annotations) and generalize it to real-world data for which obtaining labels is a cumbersome task. The problem persists even in controlled simulated environments. For example, in VR environments, a chair's visual representation can vary significantly between a game and architectural design software. In the more demanding setting of *unsupervised domain adaptation* (UDA) for classification, the source domain consists of labeled point clouds, while the target domain is completely unlabeled.

While a majority of the point cloud representation learning works have focused on improving performance in supervised and unsupervised tasks [28, 35, 40], very few have focused on the task of DA between disparate point cloud datasets. Towards addressing this challenging task, we propose a method for UDA on point cloud data for classification task, which is explained in Chapter 3. We also explore learning Wasserstein embeddings for 3D point clouds; we discuss about it next.

1.4 Need of going beyond Euclidean space

Euclidean space is the most commonly used space in learning methods. It has been realized that Euclidean space is constrained in its ability to represent complex relations and hierarchies. Yet, very limited work has been done in this direction, specifically for point clouds. Motivated by this, we propose our method for extracting 3D Wasserstein embeddings for 3D point clouds in Chapter 4. In Section 4.4, we show improvements in various downstream tasks with Wasserstein embeddings as compared to Euclidean ones.

1.5 Contributions

As mentioned in the previous sections, this thesis focuses on 3D representation learning. In particular, for point cloud domain adaptation and learning representation in Wasserstein spaces. To this end, the following are our core contributions:

1. For our first work, to the best of our knowledge, we are the first to propose the use of multimodal contrastive learning within individual domains along with optimal transport for domain alignment for 3D point cloud domain adaptation.

2. We build an end-to-end framework with two contrastive losses between 3D point cloud augmentations and between a point cloud and its 2D image projections. We also include optimal transport loss for domain alignment.

3. We perform an exhaustive empirical study on two popular benchmarks called PointDA-10 and GraspNetPC-10. Our method achieves state-of-the-art performance on GraspNetPC-10 (with $\approx$ 4-12% margin) and the best average performance on PointDA-10. Our method outperforms existing methods in the majority of cases with significant margins on challenging real-world datasets. We also conduct an ablation study and explore decision boundaries for our self-supervised contrastive and OT losses to elucidate the individual contributions of each component in our method.

4. For our second work, to the best of our knowledge, we are the first to propose the use of OT metric as a distance measure in contrastive learning for point cloud data. Unlike, Euclidean embeddings, we represent a point cloud as a discrete distribution in the embedding space.

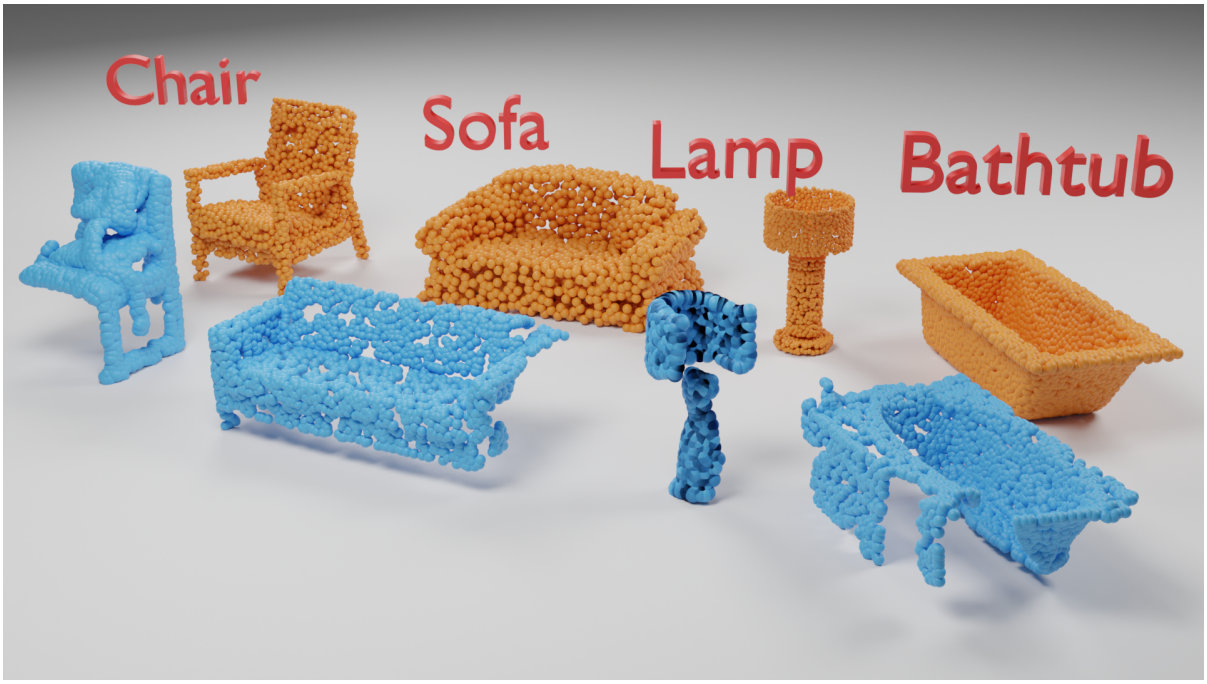


Figure 1.3: Point cloud samples from PointDA-10 dataset. Point clouds in orange are synthetically generated (ModelNet), and point clouds in blue are taken from real-world scans (ScanNet). **The figure shows the difference in point cloud geometry that arises between synthetic and real-world domains.**

5. Using this representation, we design a framework for learning Wasserstein embeddings for 3D point clouds endowed by contrastive learning with Sliced Wasserstein distances.

6. We perform exhaustive experiments on a wide variety of downstream tasks over four popular datasets to validate the effectiveness of these embeddings over Euclidean ones and other baselines. We also illustrate the efficacy of Wasserstein embeddings by visualizing the 3D features captured by the model.

1.6 Previous attempts on 3D domain adaptation

Domain adaptation task is an under-explored problem for 3D point clouds as compared to work done on it for 2D images. Recent works [1,30,34,46] propose various interesting strategies for 3D point cloud domain adaptation. [30] proposes the first benchmark dataset for point cloud domain adaptation called PointDA-10, visualizations of a few samples from the dataset are shown in Figure 1.3. This dataset consists of 3D objects/scenes extracted from 3 popular datasets, namely, ModelNet [41], ShapeNet [4], and ScanNet [9]. Among these, ModelNet and Shapenet are synthetic datasets, while ScanNet is built

from real-world scans. This gives a total of 6 possible source-target domain combinations for evaluation. [30] also proposes a method called PointDAN, where the idea is to find global alignment as well as local alignments between point clouds. [1] proposes a method called DefRec, that mainly has two task heads. The first task head has a supervised classification branch for only the source domain. It also leverages the advantages of the PointMixup [6] idea. The second head contains a self-supervised branch that is mainly responsible for extracting similar features for the same classes across domains and takes both domains as input. As a self-supervision task, they propose to deform some local region from the point cloud and then reconstruct back the original point cloud. Along similar directions, [46] propose a method called GAST, which has a supervised branch for the source domain and two self-supervised branches for both domains. As the self-supervision tasks, GAST includes point cloud rotation angle prediction to bring the point cloud into an upright orientation and distortion local prediction. Along with these branches, it also includes the SPST strategy that is explained in Section 3.2.3. [34] proposes a new dataset called GraspNet-10 mainly to evaluate adaptation performance from synthetic-real and real-real domains. The dataset includes point clouds from synthetic and two real-world domains. The real-world scans are acquired from two different sensors, namely Kinect and Realsense. [34] uses learning geometry-aware implicit representation as the self-supervision task for domain adaptation.

These works mainly focus on designing a self-supervision task to improve domain adaptation. However, we take a different approach, where along with our self-supervision task, we also optimize for explicit alignment of domains. Our idea and method are explained in detail in Chapter 3.

1.7 Organization of the Thesis

The rest of the thesis is organised as follows:

1. In Chapter 2, we commence with a succinct introduction to Optimal Transport, followed by an examination of diverse distance metrics derived from it. These metrics hold significance for the upcoming chapters.

2. In Chapter 3, we propose our method for point cloud domain adaptation that uses contrastive learning and optimal transport (COT). Further, we show evaluation on multiple datasets and show the effectiveness of our method by achieving state-of-the-art performance on GraspNetPC-10.

3. In Chapter 4, we explore learning Wasserstein embeddings for 3D point clouds. We propose a contrastive framework and show improvements in multiple downstream tasks as compared to Euclidean metrics.

Chapter 2

Optimal Transport and Wasserstein Distance

In this chapter, we first have a brief introduction to Optimal Transport and then discuss various distance metrics derived using it, which are useful for the next chapters.

A famous example used to informally explain Optimal Transport (OT) given by the French mathematician Gaspard Monge is: An individual equipped with a shovel is tasked with relocating a large pile of sand situated at a construction site. The objective for the worker is to assemble the sand into a designated configuration or a target pile, such as that of a sandcastle. Instinctively, the worker aims to reduce his overall efforts, measured, for example, by the total distance or time invested in transporting shovel loads of sand.

Mathematicians interested in OT frame this problem as a comparison between two probability distributions—essentially, two distinct sand piles with identical volumes. They examine all of the many potential ways to transform, *transport*, or modify the initial pile into the second one, and assign a *global* cost to each of these transportation scenarios, using a *local* perspective of how expensive moving an individual grain of sand from one location to another would be. Mathematicians are interested in the characteristics of the most economical transportation (least costly transport) and developing efficient computational (least time/computation complexity) techniques for it. The minimal cost not only establishes a distance metric between distributions but also imparts a complex geometric structure on the space of probability distributions. Instead of moving one item from a source to a target location, the problem of moving several items (or a distribution) to a target configuration has concrete connections with our daily lives. The process of transportation, whether it pertains to individuals, goods, or information, rarely revolves around the movement of a singular item. In significant economic challenges such as logistics, production planning, or network routing, the focal point is the movement of distributions rather than isolated entities. Tolstoy [1930], Hitchcock [1941], and Kantorovich [1942] initially worked on optimal transport driven by practical considerations. However, it wasn't until the post-1980s era, particularly with the contributions of Brenier [1991] and others, that mathematicians recognized the theory's profound potential for research. This discovery unveiled intricate connections to convexity, partial differential equations, and statistics. As the new millennium approached, professionals in computer science, imaging, and broader data sciences grasped the transformative capabilities of optimal transport

theory. It provided robust tools for studying distributions in a distinct and more abstract context, particularly in comparing readily available distributions in the form of bags-of-features or descriptors. In the coming chapters, we explore the power of optimal transport and design novel methods for 3D representation learning and show its advantage in different learning settings and various downstream tasks. We next explain the different metrics that we use in our methods derived using OT.

2.1 p-Wasserstein Distance

Optimal transport offers a way to compare two probability distributions irrespective of whether the measures have common support. It aims to find the most efficient way of transferring mass between two probability distributions, considering the underlying geometry of the probability space. Formally, given two probability distributions μ and ν on a metric space $\mathcal{X}$, for $p \geq 1$, the p-Wasserstein distance [37] is given by

$$W_p(\mu, \nu) = \left(\inf_{\pi \in \Pi(\mu,\nu)} \int_{\mathcal{X}\times\mathcal{X}} c(x, y)^p d\pi(x, y) \right)^{1/p} \tag{2.1}$$

where π is a transport plan that defines a flow between mass from μ to locations in ν, $\Pi(\mu, \nu)$ is the joint probability distribution with the marginals μ and ν and $c(x, y)$ is the ground metric which assigns a cost of moving a unit of mass $x \in \mathcal{X}$ from μ to some location $y \in \mathcal{X}$ in ν.

For the discrete case, given two discrete distributions $\hat{\mu} = \sum_{i=1}^{m} a_i \delta(x_i)$ and $\hat{\nu} = \sum_{j=1}^{n} b_j \delta(y_j)$, where $\{a_i\}_{i=1}^{m}$ and $\{b_j\}_{j=1}^{n}$ are the probability masses that should sum to 1, $\{x_i\}_{i=1}^{m}$ and $\{y_j\}_{j=1}^{n}$ are the support points in $\mathbb{R}^d$ with m and n being the number of points in each measure. The discrete form of the above equation can be given as

$$W_p(\hat{\mu}, \hat{\nu}) = \left(\min_{\psi \in U(a,b)} \langle C^p, \psi \rangle_F \right)^{1/p} \tag{2.2}$$

where $\langle \cdot, \cdot \rangle_F$ denotes the Frobenius dot-product, $C^p \in \mathbb{R}_+^{m\times n}$ is the pairwise ground metric distance, ψ is the coupling matrix and U is the set of all possible valid coupling matrices, i.e. $U(a, b) = \{\psi \in \mathbb{R}^{m\times n} : \psi \mathbb{1}_n = a, \psi^\top \mathbb{1}_m = b\}$.

2.2 Sliced-Wasserstein Distance

Interestingly, there exists a closed-form solution for Wasserstein distance only when the distributions are one-dimensional measures with L^p norm as the cost function. The closed-form for Wasserstein distance in 1-D is [27]

$$W_p(\mu, \nu) = \left(\int_0^1 |F_\mu^{-1}(t) - F_\nu^{-1}(t)|^p dt \right)^{1/p} \tag{2.3}$$

where, F_μ^{-1} and F_ν^{-1} are the inverse cumulative distribution functions of μ and ν.

Generally, we are more interested in dimensions greater than one. Thus, we cannot use this closed-form solution directly to solve the OT problem efficiently. Instead, the Wasserstein distance between two measures on $\mathbb{R}^d$ can be approximated by aggregating the 1-D Wasserstein distance between their projections over multiple directions on a unit sphere, which is called the Sliced Wasserstein distance [27]:

$$SW_p(\mu, \nu) = \left(\int_{S^{d-1}} W_p(P_{\theta,\#}\mu, P_{\theta,\#}\nu)^p d\theta \right)^{1/p} \tag{2.4}$$

where, $S^{d-1} = \{\theta \in \mathbb{R}^d : \|\theta\| = 1\}$ is the d-dimensional unit sphere and $P_\theta : \mathbb{R}^d \to \mathbb{R}$ is the projection. Since the projections are now 1-D measures, we can use the closed-form solution given by Equation 2.3. When $m = n$, the Sliced Wasserstein distance can be easily computed by simply sorting points in 1-D measures and can be given by:

$$SW_p(\hat{\mu}, \hat{\nu}) = \left(\frac{1}{D} \sum_{k=1}^{D} \sum_{i=1}^{m} |x_{\alpha_{\theta_k}(i)} - y_{\beta_{\theta_k}(i)}|^p \right)^{1/p} \tag{2.5}$$

where, α_{θ_k} and β_{θ_k} are the permutation ordering in the increasing order of the support points projected to the direction θ_k with D being the total number of slicing directions. In Chapter 4, Figure 4.2 pictorially demonstrates the computation of Sliced-Wasserstein distance.

2.3 Summary

In this chapter, we first introduced the concept of Optimal Transport and saw how this theory evolved from solving basic daily practical problems to helping in scientific and important research problems. We then explained two distance metrics, them being p-Wasserstein and Sliced-Wasserstein distance. We use these distance measures in the coming chapters to build our method for 3D representation learning. In the next chapter, we explain our method for 3D point cloud domain adaptation, where the main idea is to synergize the working of contrastive learning and optimal transport.

Chapter 3

Synergizing Contrastive Learning and Optimal Transport for 3D Point Cloud Domain Adaptation

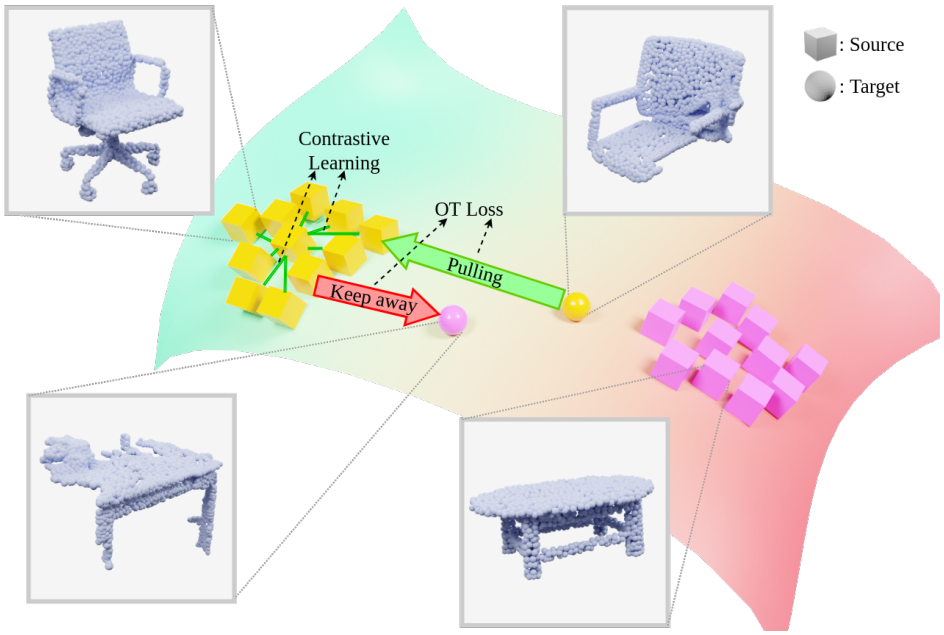


Figure 3.1: **Overview of our method for UDA.** Contrastive learning (CL) and optimal transport (OT) are designed to complement each other synergistically. CL establishes class clusters, while OT aligns objects across domains. The colors of data points denote different classes.

In this chapter, we first motivate using optimal transport and multimodal inputs for 3D point cloud domain adaptation (Section 3.1). Then explain our proposed method COT (Section 3.2). We then explain the experimental setup (Section 4.3), quantitative results (Section 3.4), and lastly present extensive ablation studies (Section 3.5).

3.1 The Motivation of using Optimal Transport and Multimodal Inputs

While a majority of the point cloud representation learning works have focused on improving performance in supervised and unsupervised tasks [28, 35, 40], very few have focused on the task of *domain adaptation* between disparate point cloud datasets. Recent works focus on incorporating self-supervised learning (SSL) approaches to learn similar features for both domains, along with a regular source domain supervision [1, 34, 46]. [1] introduces a self-supervised approach based on deformation reconstruction and leverages PointMixup [6]. [46] learns a domain-shared representation of semantic categories by leveraging two self-supervised geometric learning tasks as feature regularizers. [34] proposes a self-supervised task of learning geometry-aware implicits for domain-specific variations. These works mainly focus on designing a self-supervision task that can improve domain adaptation. The point clouds belonging to the same class must not only be closer in each individual domain but also achieve cross-domain alignment. However, our analysis reveals that explicit cross-domain alignment is under-explored, given the significant margins between classification accuracies on source and target domains.

Based on our aforementioned observations, we draw inspiration from recent SSL contrastive learning research [2, 5, 20], which has enjoyed major success in other domains such as image and text. We propose a **C**ontrastive SSL method on point clouds to improve class separation individually in both source and target domains that share a common label space. In addition, optimal transport (OT) based methods [10] have also shown promising results as they jointly learn the embeddings between both domains by comparing their underlying probability distributions and exploiting the geometry of the feature space. Thus, we employ **OT** to achieve better cross-domain alignment for domain adaptation. Figure 3.1 provides a visual overview of our method (COT).

To further reduce the domain shift and learn high quality transferable point cloud embeddings, we also leverage the idea of multi-modality within the source and target domains. As another source of information, we use multiple 2D projections of a 3D point cloud. 2D projections aim to incorporate correspondence understanding between 3D and 2D features for a shape. It explains the proficiency of humans to map visual concepts from 2D images to understand the 3D world [31]. The idea is to learn a unified space where a shape's 2D features and transformation-agnostic 3D features are aligned. This multimodal learning of 2D-3D features assists the network in extracting reliable discriminative features and positively aids domain adaptation performance. Also, 2D projections can be seen as a form of data augmentation for point clouds introducing more diversity in the training data and promoting better generalization. 2D projections from various viewpoints allow capturing *silhouette* and *surface boundary* information for shape understanding that is harder to derive from just point-wise distances. Moreover, mature 2D CNNs provide valuable features which when aggregated and combined with 3D features, provide informative global shape descriptors for the 3D point cloud.

Motivated by the aforementioned observations, we design an end-to-end framework that consists of a multimodal self-supervised contrastive learning setup for both source and target domains individually and OT loss for domain alignment. We explain our method for UDA next.

3.2 Contrastive (+) Optimal Transport (COT)

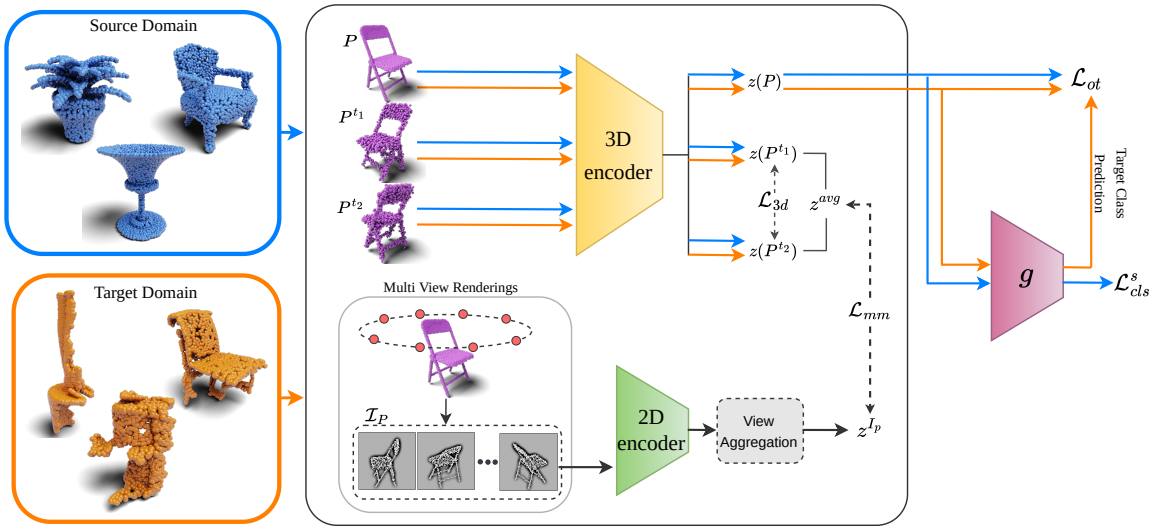


Figure 3.2: **Overview of our framework.** Three main components: self-supervised contrastive training ($\mathcal{L}_{3d}$, $\mathcal{L}_{mm}$), self-supervised OT training between both domains ($\mathcal{L}_{ot}$) and a supervised training on source domain ($\mathcal{L}_{cls}$). Contrastive loss uses features from shared point cloud and image encoders with point cloud augmentations and 2D image projections. OT and classifier losses takes features of original point cloud samples from shared point cloud encoder.

This section describes our method for UDA of point clouds for classification task. Our method is endowed by *multimodal self-supervised contrastive learning and OT for domain alignment*. The self-supervised multi-modal contrastive learning module leverages both, the 3D information and their corresponding 2D image projections of point clouds. It produces initial class clusters in the source and target domains individually. Subsequently, our OT module better aligns the same class clusters across domains. We additionally also train a classifier on the source domain to improve the class separation, which in turn lessens the burden on our adaptation module. Our setup aims at learning high quality embeddings, jointly for source and target domains, by exploiting both *contrastive learning with augmentations* and the *multimodal information of the input point clouds*, while simultaneously *reducing the domain shift* across the domains. Our architecture is illustrated in Figure 3.2.

Let a point cloud $P = \{x_1, \ldots, x_n\}$, where $x_i \in \mathbb{R}^3$, be a set of 3D points of cardinality n. Let $\mathcal{D}^s = \{P_i^s, y_i\}_{i=1}^{n_s}$ denote the *labeled source domain dataset*, where P_i^s denotes the i-th source point cloud and y_i its associated class label that takes values in $\mathcal{Y} = \{1, \ldots, K\}$. Note that $\mathcal{Y}$ is a set of shared class labels that is *common* to both the source and target domains. The *target domain dataset* $\mathcal{D}^t = \{P_i^t\}_{i=1}^{n_t}$ contains unlabeled point clouds. The cardinality of $\mathcal{D}^s$ and $\mathcal{D}^t$ are n_s and n_t respectively.

Then, the task of UDA for point cloud classification boils down to learning a *domain invariant function* $f : \mathcal{P} \rightarrow \mathcal{Y}$, where $\mathcal{P}$ is a union of unlabeled point clouds from both $\mathcal{D}^s$ and $\mathcal{D}^t$.

3.2.1 Self-Supervised Contrastive Learning

Motivated by the advancement of contrastive learning [5, 20], where the goal is to pull samples from common classes closer in the embedding space, we build a method to extract 3D and 2D features of point clouds and fuse this information to form initial domain class clusters.

We employ a contrastive loss between augmented versions of a point cloud, which we term as a *3D-modal association loss*, to learn similar features for samples from the same class. This loss forces the point cloud learning to be invariant to geometric transformations. Additionally, we introduce a contrastive loss between the 3D point cloud features and their corresponding projected 2D image features, termed as *multi-modal association loss*. The intuition behind this multi-modal loss is to take advantage of the rich multi-view latent 2D information inherent in the 3D point clouds. Next, we explain these components in detail.

3.2.1.1 3D-modal association loss

Let P_b be a point cloud from a randomly drawn batch B of size k from either $\mathcal{D}^s$ or $\mathcal{D}^t$. Given a set of affine transformations T, we generate two augmented point clouds $P_b^{t_1}$ and $P_b^{t_2}$, where t_1 and t_2 are compositions of transformations picked randomly from T. Additionally, we use random point dropout and add random noise to each point in a point cloud individually to introduce object surface distortions. These transformations introduce geometric variations, which are then used to curate samples that serve as positive pairs. The augmented point clouds $P_b^{t_1}$ and $P_b^{t_2}$ are then mapped to a d-dimensional feature space using a 3D encoder function producing embeddings $z(P_b^{t_1})$ and $z(P_b^{t_2})$, respectively. These embeddings serve as *positive pairs* and therefore our objective is to place them closer to one another in the feature space.

We define the similarity between the i-th embedding transformed by t_x and the j-th embedding transformed by t_x, with $x \in \{1, 2\}$, as given in Eqn 3.1

$$\langle (i, t_x), (j, t_x) \rangle_{ST} = \exp\left(s(z(P_i^{t_x}), z(P_j^{t_x}))/\tau \right) \tag{3.1}$$

where s denotes the cosine-similarity function and τ is the temperature hyperparameter.
Our 3D-modal association loss is then given as shown in Eqn 3.2

$$\mathcal{L}_{3d} = -\log \left\{ \frac{\langle (i, t_1), (i, t_2) \rangle_{ST}}{\sum_{j=1}^{k} \langle (i, t_1), (j, t_1) \rangle_{ST} + \sum_{j=1}^{k} \langle (i, t_1), (j, t_2) \rangle_{ST}} \right\} \tag{3.2}$$

For both source and target, we randomly draw respective batches and perform 3D-modal association separately. This method of self-supervised contrastive learning generates class clusters in both domains

individually and has been shown to be useful especially for the target domain, as its supervision signal is missing. We further guide the feature learning by introducing image modality in the optimization. We explain our multi-modal association loss next.

3.2.1.2 Multi-modal association loss

We consider using point cloud projections in our method, as the image features can provide another level of discriminative information. 2D projections from various viewpoints allow capturing *silhouette* and *surface boundary* information for shape understanding that is harder to derive from just point-wise distances. Breaking away from the common way of fusing multimodal information [3, 18] where the embeddings of two modalities are fused by simply concatenating or averaging them, we instead compute associative losses between 3D features and image features to establish 2D-3D correspondence understanding helping to provide informative global representation.

As contrastive learning is known to be good for alignment tasks, we advocate using a contrastive objective to fuse multimodal (3D and 2D) information. Let $\mathcal{I}_P = \{I_n\}_{n=1}^m$ be the set of m 2D image projections of point cloud P. To generate these images, we set virtual cameras around the object in a circular fashion to obtain views of the object from all directions. For a point cloud P, each of its corresponding 2D images is passed to a 2D encoder, generating a d-dimensional embedding. Following [15,35], we use a simple max-pooling operation to aggregate feature information from all views and get a d-dimensional vector z^{I_P}. In order to fuse the 3D augmented point cloud embeddings (i.e., $z(P^{t_1})$ and $z(P^{t_2})$) with the 2D point cloud embedding z^{I_P}, we compute the average of the 3D augmented point cloud embeddings to get z^{avg}. We then use the z^{avg} and z^{I_P} that contain summarized information from 3D and 2D modalities respectively in a self-supervised contrastive loss to maximize their similarity in the embedding space. We define the similarity between the i-th embedding z_i and the j-th embedding z'_j as $\langle z_i, z'_j \rangle_S = \exp\Big(s(z_i, z'_j)/\tau\Big)$. Then, our multi-modal association loss is given by

$$\mathcal{L}_{mm} = -\log\left\{\frac{\langle z_i^{avg}, z_i^{I_P}\rangle_S}{\sum_{j=1}^{k}\langle z_i^{avg}, z_j^{avg}\rangle_S + \sum_{j=1}^{k}\langle z_i^{avg}, z_j^{I_P}\rangle_S}\right\} \tag{3.3}$$

The total self-supervised contrastive loss is given by adding the 3D-modal association loss ($\mathcal{L}_{3d}$) that maximizes the similarity between augmentations of a point cloud and the multi-modal association loss ($\mathcal{L}_{mm}$) that maximizes the similarity between 3D and 2D features of a point cloud.

3.2.2 Domain Alignment via Optimal Transport

As explained in Section 3.2.1, contrastive learning generates class clusters in source and target domains individually. The underlying idea is to further achieve alignment of point clouds belonging to the same class across two domains. We leverage an OT based loss that uses point cloud features and

source labels for domain alignment. The classifier $g : \mathbb{R}^d \rightarrow \mathcal{Y}$ that maps the point cloud embedding from feature space to label space also needs to work well for the target domain. The OT flow is greatly dependent on the choice of the cost function as shown by [8]. Here, as we want to jointly optimize the feature and the classifier decision boundary learning, we define our cost function as

$$\sum_{i=1}^{k}\sum_{j=1}^{k} c(z_i^s, z_j^t) = \alpha||z_i^s - z_j^t||_2^2 + \beta||y_i^s - g(z_j^t)||_2^2 \tag{3.4}$$

where superscripts s and t denote the source and target domains, respectively. α, β are the weight coefficients. Here, the first term computes the squared-$L2$ distance between the embeddings of source and target samples. The second term computes squared-$L2$ distance between the classifier's target class prediction and the source ground truth label. Jointly, these two terms play an important role in pulling or keeping apart source and target samples for achieving domain alignment. For example, if a target sample lies far from a source sample having the same class, the first term would give a high cost. However, for a decently trained classifier, the distance between its target class prediction and source ground truth label would be less, thus making the second term low. This indicates that these source and target samples must be pulled closer. Conversely, if a target sample lies close to a source sample having a different class, the first term would be low, and the second term would be high, indicating this sample should be kept apart. As evident from the example, the second term is a guiding entity for inter-domain class alignment. It penalizes source-target samples based on their classes and triggers a pulling mechanism. The problem of finding optimal matching can be formulated as $\psi^* = \min_{\psi \in U(a_s, b_t)} \langle C^p, \psi \rangle_F$, where ψ^* is the ideal coupling matrix, a_s and b_t are the uniform marginal distributions of source and target samples from a batch. The optimal coupling matrix ψ^* is computed by freezing the weights of the 3D encoder function and the classifier function g. The OT loss for domain alignment is given by

$$\mathcal{L}_{ot} = \sum_{i=1}^{k}\sum_{j=1}^{k} \psi_{ij}^*(\alpha||z_i^s - z_j^t||_2^2 + \beta\mathcal{L}_{ce}(y_i^s, g(z_j^t)) \tag{3.5}$$

where $\mathcal{L}_{ce}$ is the cross-entropy loss.

3.2.3 Overall Training Loss

The overall pipeline of our unsupervised DA method is trained with the combination of the following objective functions $\mathcal{L}_{total} = \mathcal{L}_{3d} + \mathcal{L}_{mm} + \mathcal{L}_{ot} + \mathcal{L}_{cls}^s$. The loss consists of three self-supervised losses (i.e., $\mathcal{L}_{3d}$, $\mathcal{L}_{mm}$ and $\mathcal{L}_{ot}$) and a supervised loss $\mathcal{L}_{cls}^s$. Besides three SSL tasks, supervised learning is performed based on source samples and labels. For this purpose, a regular cross-entropy loss or a *mixup* variant can be applied [44]. We use a supervised loss ($\mathcal{L}_{cls}^s$) inspired by the PointMixup method (PCM) [6]. PCM is a data augmentation method for point clouds by computing interpolation between samples. Augmentation strategies have proven to be effective and enhance the representation capabilities of the model. Similarly, PCM has shown its potential to generalize across domains and robustness to noise and geometric transformations.

We also employ the *self-paced self-training* (SPST) strategy introduced by [46] to improve the alignment between domains. In SPST, pseudo-labels for the target samples are generated using the classifier's prediction and confidence threshold. The first step computes the pseudo labels for the target samples depending on the confidence of their class predictions, while the next step updates the point cloud encoder and classifier with the computed pseudo labels for target and ground truth labels of source. In our method, we use SPST strategy as a fine-tuning step for our models.

3.3 Experimental Setup

In this section, we explain our baselines, all the used datasets, and the implementation details with the used hyperparameters.

3.3.1 Baslines

We conduct an exhaustive experimental study to show the effectiveness of the learned representations and the significance of our COT. We consider recent state-of-the-art self-supervised methods such as DANN [14], PointDAN [30], RS [32], DefRec+PCM [1], GAST [46] and ImplicitPCDA [34] for comparison. Additionally, we report results for the baseline without adaptation (unsupervised) which trains the model using labels from the source domain and tests on the target domain. The supervised method is the upper bound which takes labels from the target domain into consideration during training.

3.3.2 Datasets

3.3.2.1 PointDA-10

[30] introduces PointDA-10, which is a combination of ten common classes from ModelNet [40], ShapeNet [4] and ScanNet [9]. ModelNet and ShapeNet are synthetic datasets sampled from 3D CAD models, containing $4,183$ training, 856 test samples and $17,378$ training, $2,492$ test samples, respectively. On the other hand, ScanNet consists of point clouds from scanned and reconstructed real-world scenes and consists of $6,110$ training and $1,769$ test samples. Point clouds in ScanNet are usually incomplete because of occlusion by surrounding objects in the scene or self-occlusion in addition to realistic sensor noises. We follow the standard data preparation procedure used in [1, 30, 34, 46].

3.3.2.2 GraspNetPC-10

This dataset is proposed by [34], which consists of synthetic and real-world point clouds for ten object classes. It is developed from GraspNet [11] by re-projecting raw depth scans to 3D space and applying object segmentation masks to crop out the corresponding point clouds. Raw depth scans are captured by two different depth cameras, Kinect2 and Intel Realsense to generate real-world point clouds. In the

Synthetic, Kinect, and RealSense domains, there are $12,000$ training, $10,973$ training, $2,560$ testing, and $10,698$ training, $2,560$ testing point clouds, respectively. There exist different levels of geometric distortions and missing parts. Unlike PointDA-10, point clouds in GraspNetPC-10 are not aligned and all domains have almost uniform class distribution.

3.3.3 Implementation Details

For data pre-processing, we follow [1] and align the positive Z axis of all point clouds from the whole PointDA-10 dataset. We use the farthest point sampling algorithm to sample 1024 points uniformly across the object surface. Further, all the point clouds are normalized and scaled to fit in a unit-sphere. For getting renderings of point clouds from multiple views, we place orthographic cameras in a circular rig. We set the number of views to 12 and the image size as 224×224. We set points color to $white$, background color to $black$, points radius to 0.008, and points per pixel to 2. For getting two augmented versions of the original point cloud used in self-supervised contrastive learning, we compose spatial transformations picked from random point cloud scaling, rotation, and translation. The original point cloud which is passed to the source classifier is only transformed with random jittering and random rotation about its Z axis.

For a fair comparison with recent works [1, 34, 46] we use DGCNN [39] as our 3D encoder for extracting global point cloud features. We choose a pre-trained ResNet-50 [17] as our image feature extractor. Both 3D and 2D encoders embed their respective modality into a 256 dimensional feature space for contrastive learning. Whereas, for the classification task, 1024 dimensional feature vector of the original point cloud is used. We use a 3-layer MLP as our classifier, having $(512, 256, 10)$ neurons respectively. Please note that for testing the classification performance, we do not use 2D features and only use the global features given by the 3D encoder. We set the temperature parameter τ used in contrastive losses $\mathcal{L}_{3d}$ (Eqn 3.2) and $\mathcal{L}_{mm}$ (Eqn 3.3) as 0.1. To solve the optimization problem of the optimal coupling matrix, we use the POT library [12]. We do a grid search to find the best α and β combination from $\mathcal{L}_{ot}$. For most of the dataset combinations, we set the hyperparameters α and β to 0.001 and 0.0001, respectively.

We perform all our experiments on NVIDIA RTX-2080Ti GPUs using the Pytorch framework for implementing our models. We set the batch size to 32, learning rate as 0.001 with cosine annealing as the learning rate scheduler and use Adam optimizer. We set weight decay to 0.00005 and momentum to 0.9. In total, we train our models for 150 epochs on PointDA-10 and 120 epochs on GraspNetPC-10 dataset. We report results from the model with the best classification accuracy on source validation set, as target labels are unavailable.

3.4 Unsupervised DA: Classification

In Tables (3.1, 3.2), we compare the results of our COT with the existing point cloud domain adaptation methods [1, 30, 34, 46] on PointDA-10 and GraspNetPC-10 datasets respectively. Similar to [34] and [46], we also test our methodology with SPST strategy. As shown in Table 3.1, COT achieves SoTA performance in terms of the overall average performance on PointDA-10 dataset. We observe that COT beats existing methods by a huge margin when the target dataset is synthetic. This is because target point clouds have well-defined geometry, and the classifier can make accurate predictions with high confidence, thus majorly helping alignment. As existing methods only propose to use self-learning tasks, their performance is very low compared to our self-learning task with explicit domain alignment endowed by OT. For the settings where the target dataset is real, it becomes harder for the classifier to provide good predictions, making the alignment process noisy. In these settings, we achieve on-par results compared to the existing methods. In $S \rightarrow M$, our method with SPST strategy outperforms existing methods $\approx 8\%$, and in $M \rightarrow S$, we achieve on-par results compared to the existing methods. For PointDA-10, we observe that when the target domain is synthetic, the learned features are distinctive; however, when the target domain is real, the features lack distinctive power. This portrays the challenging setting of synthetic to real adaptation. Overall we achieve the highest average accuracy on PointDA-10 dataset showing effectiveness of COT.

Our method outperforms all the existing methods with a significant margin on all the combinations of GraspNetPC-10 dataset, as shown in Table 3.2. COT beats existing methods in both with and without SPST strategy; also, in some cases, it beats the supervised method (upper bound). It is interesting to note the difference in behaviour of COT and other methods on real-world data in PointDA-10 and GraspNetPC-10. PointDA-10, in general, has a very skewed class-wise sample distribution and has a small set of real-world samples. Whereas, GraspNetPC-10 has almost uniform class-wise sample distribution with approximately double the size of ScanNet. COT performs significantly better with larger datasets and almost equal class-wise sample distribution. Existing methods that propose classification-based [46] or geometry-aware implicit learning-based [34] tasks fall short in terms of performance boost compared to COT when real-world datasets are large and have uniform class distribution. This shows the effectiveness of COT for unsupervised domain adaptation achieving SoTA performance on real-world data from GraspNet-10 dataset.

3.4.1 Domain Alignment

In this section, we discuss our used sampling strategy for creating a batch and explain its working in our $\mathcal{L}_{ot}$ loss for domain alignment. For every iteration, we use random sampling to draw source and target batches independently. Note that it does not ensure the coherence of source and target classes in a batch. Using these batches, the OT flow finds the best one-to-one matching amongst both domains using the defined cost function and updates both network's (encoder and classifier) weights to minimize the $\mathcal{L}_{ot}$ loss. Even though we use random sampling we find that repeating this process for multiple

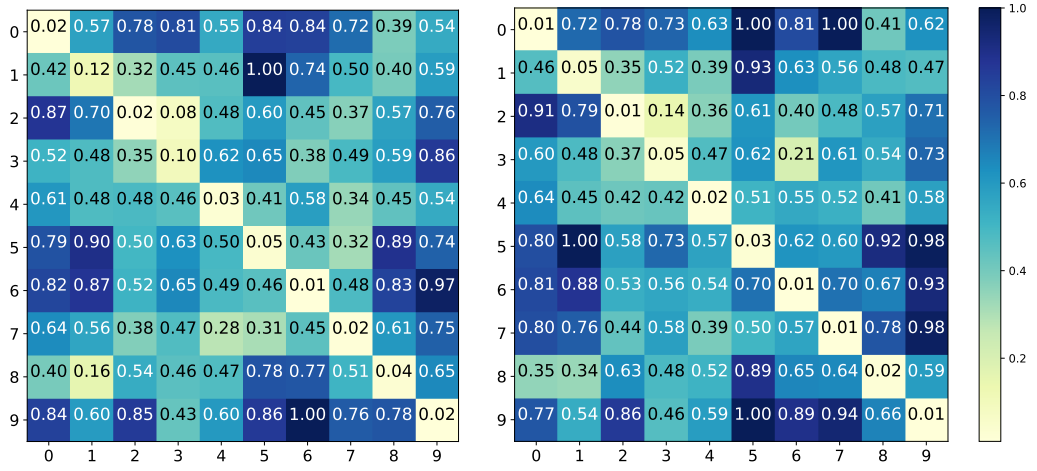
Methods	SPST	M → S	M → S*	S → M	S → S*	S* → M	S* → S	Avg.
Supervised		93.9 ± 0.2	78.4 ± 0.6	96.2 ± 0.1	78.4 ± 0.6	96.2 ± 0.1	93.9 ± 0.2	89.5
Baseline(w/o adap.)		83.3 ± 0.7	43.8 ± 2.3	75.5 ± 1.8	42.5 ± 1.4	63.8 ± 3.9	64.2 ± 0.8	62.2
DANN [14]		74.8 ± 2.8	42.1 ± 0.6	57.5 ± 0.4	50.9 ± 1.0	43.7 ± 2.9	71.6 ± 1.0	56.8
PointDAN [30]		83.9 ± 0.3	44.8 ± 1.4	63.3 ± 1.1	45.7 ± 0.7	43.6 ± 2.0	56.4 ± 1.5	56.3
RS [32]		79.9 ± 0.8	46.7 ± 4.8	75.2 ± 2.0	51.4 ± 3.9	71.8 ± 2.3	71.2 ± 2.8	66.0
Defrec+PCM [1]		81.7 ± 0.6	51.8 ± 0.3	**78.6** ± 0.7	54.5 ± 0.3	73.7 ± 1.6	71.1 ± 1.4	68.6
GAST [46]		83.9 ± 0.2	**56.7** ± 0.3	76.4 ± 0.2	55.0 ± 0.2	73.4 ± 0.3	72.2 ± 0.2	69.5
	✓	84.8 ± 0.1	**59.8** ± 0.2	80.8 ± 0.6	56.7 ± 0.2	81.1 ± 0.8	74.9 ± 0.5	73.0
ImplicitPCDA [34]		**85.8** ± 0.3	55.3 ± 0.3	77.2 ± 0.4	**55.4** ± 0.5	73.8± 0.6	72.4 ± 1.0	70.0
	✓	**86.2** ± 0.2	58.6 ± 0.1	81.4 ± 0.4	**56.9** ± 0.2	81.5 ± 0.5	74.4 ± 0.6	73.2
COT		83.2 ± 0.3	54.6 ± 0.1	78.5 ± 0.4	53.3 ± 1.1	**79.4** ± 0.4	**77.4** ± 0.5	**71.0**
	✓	84.7 ± 0.2	57.6 ± 0.2	**89.6** ± 1.4	51.6 ± 0.8	**85.5** ± 2.2	**77.6** ± 0.5	**74.4**

Table 3.1: Classification accuracy (%) on the PointDA-10. M: ModelNet, S: ShapNet, S*: ScanNet; → indicates the adaptation direction. SPST: self-paced self-training. Results in black and blue represent accuracy without and with SPST strategy, respectively. Bold represents the best result and underlined represents the second best for both the colors.

Methods	SPST	Syn. → Kin.	Syn → RS.	Kin. → RS.	RS. → Kin.	Avg.
Supervised		97.2 ± 0.8	95.6 ± 0.4	95.6 ± 0.3	97.2 ± 0.4	96.4
Baseline(w/o adap.)		61.3 ± 1.0	54.4 ± 0.9	53.4 ± 1.3	68.5 ± 0.5	59.4
DANN [14]		78.6 ± 0.3	70.3 ± 0.5	46.1 ± 2.2	67.9 ± 0.3	65.7
PointDAN [30]		77.0 ± 0.2	72.5 ± 0.3	65.9 ± 1.2	82.3 ± 0.5	74.4
RS [32]		67.3 ± 0.4	58.6 ± 0.8	55.7 ± 1.5	69.6 ± 0.4	62.8
Defrec+PCM [1]		80.7 ± 0.1	70.5 ± 0.4	65.1 ± 0.3	77.7 ± 1.2	73.5
GAST [46]		69.8 ± 0.4	61.3 ± 0.3	58.7 ± 1.0	70.6 ± 0.3	65.1
	✓	81.3± 1.8	72.3 ± 0.8	61.3 ± 0.9	80.1 ± 0.5	73.8
ImplicitPCDA [34]		81.2 ± 0.3	73.1 ± 0.2	66.4 ± 0.5	82.6 ± 0.4	75.8
	✓	94.6 ± 0.4	80.5 ± 0.2	76.8 ± 0.4	85.9 ± 0.3	84.4
COT		**87.7** ± 0.7	**80.2** ± 2.1	**69.3** ± 5.2	**85.8** ± 4.3	**80.0**
	✓	**98.2** ± 0.5	**83.7** ± 0.2	**81.9** ± 2.1	**98.0** ± 0.1	**91.0**

Table 3.2: Classification accuracy (%) on the GraspNet-10 dataset. Sys.: Synthetic domain, Kin.: Kinect domain, RS.: Real domain; → indicates the adaptation direction. SPST: self-paced self-training. Results in black and blue represent accuracy without and with SPST strategy, respectively. Bold represents the best result and underlined represents the second best for both the colors.

iterations eventually converges the overall alignment loss ($\mathcal{L}_{ot}$) giving discriminative features for classes with aligned source and target distributions. For examining the distance between class clusters from the source and target, we compute the maximum mean discrepancy (MMD) between learned point cloud features. In Figure 3.3, we show class-wise MMD, where Figures 3.3a, 3.3b are for baseline (without adaptation) and our COT respectively on ShapeNet to ModelNet. The diagonal of the matrix represents MMD between the same classes from source and target, and the upper and lower triangular matrices represent MMD between different classes for source and target. It is clearly evident that the MMD matrix for our COT has higher distances in the upper and lower triangular regions than the baseline. This shows that classes within the source and target individually are well separated. Further, the diagonal values for our COT are lower than the baseline without adaptation, indicating that the same classes in source and target are closer for features obtained from our method. Overall, we can see that point cloud embeddings generated by COT have better inter-class distances and source and target class alignment.



(a) (b)

Figure 3.3: Class-wise MMD for $S \rightarrow M$ for (a) baseline (only PCM w/o adaptation), and (b) our COT with SPST.

3.4.2 Discussion: Decision Boundary

We also examine the decision boundaries of our learned models. Figure 3.4 illustrates the decision boundaries from early (top-row) and final (bottom-row) epochs for four variants of our model. For this experiment, we select target samples from the hidden space of our trained models. We consider four variants of our model, i.e., i) only PCM (no adaptation), ii) contrastive learning with PCM, iii) contrastive learning and OT with PCM (our COT method), and iv) our COT fine-tuned with SPST strategy. All the representations are retrieved with the labels predicted by our trained model. Next, we fit the SVM and consider a "one-vs-rest strategy" to visualize the decision boundaries.

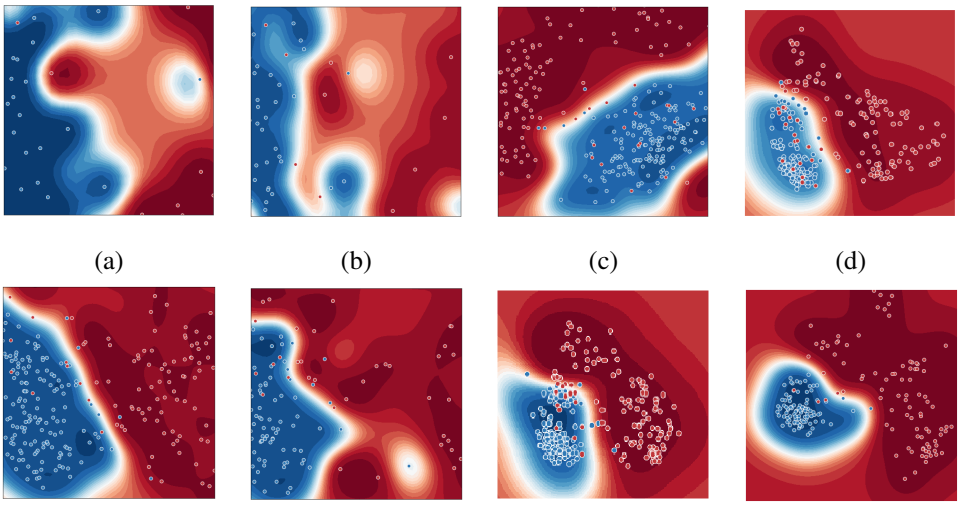


Figure 3.4: Early (top-row) and final (bottom-row) epochs decision boundaries on target samples for One-vs-Rest (Monitor class) for $S \rightarrow M$. (a), (e) Only PCM (without adaptation), (b), (f) Contrastive learning with PCM, (c), (g) Optimal transport and contrastive learning with PCM (Our COT) and (d), (h) Our COT fine-tuned with SPST.

From Figures 3.4a to 3.4d and 3.4e to 3.4h, we can clearly interpret that the baseline model with only PCM and no adaptation leads to irregular boundaries in Figures 3.4a and 3.4e. The representations are enhanced, and the boundary becomes smoother by applying contrastive learning to both the domains in Figures 3.4b and 3.4f. In contrast, training the model with our COT, which includes the previous two strategies (PCM and contrastive learning) along with OT loss further improves the decision boundaries in Figures 3.4c and 3.4g. Finally, with the SPST strategy, which finetunes the COT with pseudo labels of target samples, the region gets even more compact and smoother in Figures 3.4d and 3.4h. This shows that contrastive learning separates the two classes which are improved by OT alignment. Also, SPST further makes the classes more compact and achieves the best results.

3.5 Ablation Studies

We perform ablation studies to understand the significance of proposed losses in our method. In Table 3.3, we compare the results of our COT trained with various components on PointDA-10. $\mathcal{L}_{3d}$ is always used as it is our base self-learning task for 3D point clouds. The significance of $\mathcal{L}_{ot}$ can be seen by comparing row 3 and row 2. When $\mathcal{L}_{ot}$ is removed from COT, the performance drops on almost all settings. Comparing row 3 and row 1, we can see the effect of $\mathcal{L}_{mm}$ as the performance decreases for all settings when it is turned off. In both cases, the average accuracy also drops. This indicates positive

contribution of both $\mathcal{L}_{ot}$ and $\mathcal{L}_{mm}$ in the formulation of our COT. A similar trend is also observed with the SPST strategy as well. Comparing row 6 with rows 4 and 5, we see the best performance when both losses are used. Also, note that SPST increases the performance for all three settings shown. Overall, these results suggest that both image modality and OT-based domain alignment are crucial for achieving the best results.

$\mathcal{L}_{3d}$	$\mathcal{L}_{ot}$	$\mathcal{L}_{mm}$	SPST	M → S	M → S*	S → M	S → S*	S* → M	S* → S	Avg.
✓	✓			82.50	53.82	74.65	47.26	75.35	71.39	67.5
✓		✓		82.66	46.64	**78.50**	**53.82**	**82.24**	75.40	69.9
✓	✓	✓		**83.20**	**54.61**	**78.50**	53.30	79.44	**77.41**	**71.0**
✓	✓		✓	**84.91**	56.76	84.93	47.26	77.22	73.07	70.7
✓		✓	✓	**84.91**	54.32	85.51	**53.31**	**86.0**	75.92	73.3
✓	✓	✓	✓	84.71	**57.66**	**89.60**	51.61	85.50	**77.69**	**74.4**

Table 3.3: Ablation Study: Target classification accuracy for UDA task on PointDA-10 dataset. Bold represents best results.

3.5.1 Class-wise Performance Analysis

In this section, we analyse the class-wise accuracy of COT on PointDA-10 and and GraspNetPC-10 datasets. Results for PointDA-10 and GraspNetPC-10 are show in Tables 3.4 and 3.5, respectively

	Bathtub	Bed	Bookshelf	Cabinet	Chair	Lamp	Monitor	Plant	Sofa	Table
S*→ M	0.98	0.98	0.99	0	0.98	0.9	0.86	0.9	0.97	0.98
S*→ S	0.86	0	0.98	0.05	0.96	0.67	0.65	0.8	0.36	0.95
M→ S	0.85	0.52	0.98	0	0.94	0.65	0.84	0.97	0.92	0.93
M→ S*	0.46	0.39	0.4	0.05	0.69	0.63	0.74	0.8	0.45	0.69
S→ S*	0.54	0	0.12	0	0.7	0.73	0.77	0.32	0.45	0.58
S→ M	1	0.99	0.62	0.57	0.99	0.95	1	0.91	0.98	1

Table 3.4: Class-wise accuracies of our COT (with SPST) on the PointDA-10 dataset

In the case of PointDA-10, in almost all cases, the cabinet class is the toughest to classify. In some of the combinations, even a single example from this class is not classified correctly. In the case of GraspNetPC-10, all the samples belonging to the class Dish are always classified correctly.

	Box	Can	Banana	Drill	Scissors	Pear	Dish	Camer	Mouse	Shampoo
Syn→ Kin	1	1	0.98	0.99	0.92	1	1	1	1	1
Syn→ RS	0.95	0.97	0.28	0.84	0.96	0.69	1	1	1	0.65
Kin→ RS	1	0.78	0.63	0.98	0.9	0.31	1	0.83	0.99	0.96
Rs→ Kin	1	1	0.98	0.99	0.98	1	1	1	0.85	1

Table 3.5: Class-wise accuracies of our COT (with SPST) on the GraspNetPC-10 dataset

3.6 Summary

In this chapter, we presented a method to tackle the problem of domain adaptation on 3D point clouds for classification. We introduced a novel methodology to synergize contrastive learning and optimal transport for effective UDA. Our method focuses on reducing the domain shift and learning high-quality transferable point cloud embeddings. Our empirical study reveals the effectiveness of COT as it outperforms existing methods in overall average accuracy on one dataset, and achieves SoTA performance on another. The conducted ablation studies demonstrate the significance of our proposed method. We identify a limitation of our method, i.e., it currently assumes a fixed set of classes in both domains that limits generalizability. Also, an area where our method can be significantly improved is the domain matching cost function. The OT loss can be improved to get robust cross-domain matching. An interesting future direction would be to extend our OT-based approach for UDA of point clouds on more complex tasks like segmentation or object detection in indoor scenes. In the next chapter, we discuss about how to use optimal transport to learn effective 3D point cloud representations towards any downstream task.

Chapter 4

Going Beyond Euclidean Space for Learning Point Cloud Representations

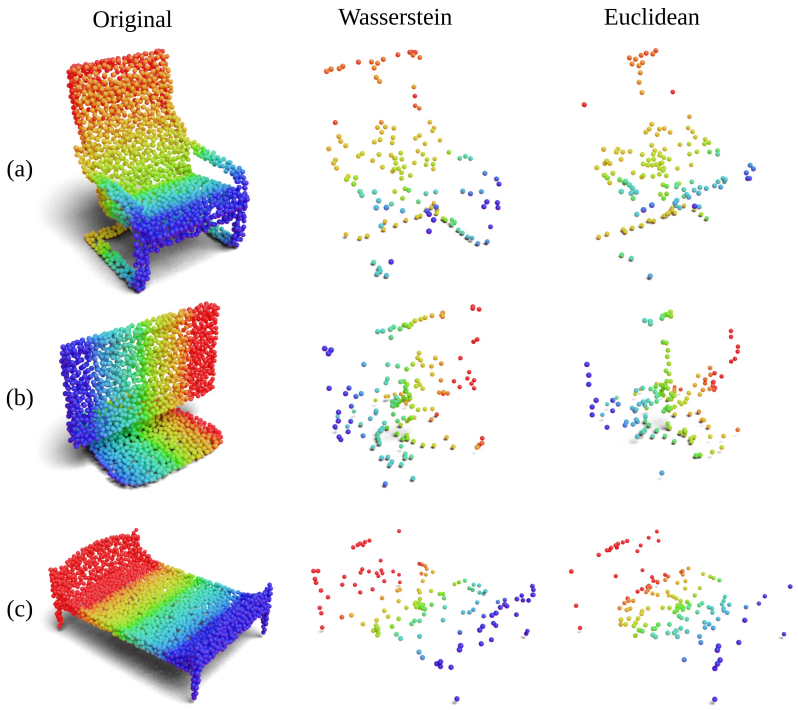


Figure 4.1: **Critical points.** The embedding of a point cloud is determined by the selected critical points. Networks trained with different distance metrics yield different sets of critical points. (a), (b) and (c) represent the original point cloud (first column), critical point set for Wasserstein space (second column), and Euclidean space (third column) for three examples (Chair, Monitor & Bed). The network trained in Wasserstein space captures better overall global geometry.

In this chapter, we first discuss the limitations of widely used Euclidean spaces (Section 4.1). We then present a method endowed by contrastive learning to learn Wasserstein embeddings for 3D point clouds (Section 4.2). We then explain the experimental setup (Section 4.3) and show experimental results on

various downstream tasks (Section 4.4). Finally, we show various ablation studies (Section 4.5) and show an interesting study of our self-explainable method by capturing critical points of point clouds better than embeddings in Euclidean space (Section 4.6).

4.1 The Motivation of Learning in Wasserstein Space

A common choice of recent 3D point cloud representation learning methods is to operate and represent point clouds as *point vectors* in Euclidean spaces. Where, for example, the relation between data points is depicted by either angle or L-2 distance. It is well known that the embedding space largely determines the quality of embeddings, as it depends on how well the *target space* can capture the underlying structure present in the data. Euclidean space is confined in its potential to capture complex structures and possible semantic relations. For example, if we consider a binary tree, the number of nodes increases exponentially as the depth of the tree increases. On the other hand, the volume of the ball in an Euclidean space grows polynomial with the radius. So, if we try to embed trees in Euclidean space, the outermost leaves will become increasingly close to one another, and we will quickly run out of space to embed the data. In contrast, in Hyperbolic space, the volume of balls grows exponentially with the radius. Thus, a tree can be embedded with very low distortion in Hyperbolic space, but this is not possible in Euclidean space. The selection of *target space* itself can give increasingly more room to embed complex hierarchical data. Realizing these drawbacks, many works use Hyperbolic space [24,25] to capture this uncertainty and asymmetric relationship for work and graph embeddings.

As Euclidean space is constrained in its ability to represent data structures, we need to *go beyond Euclidean space* to get more expressive embeddings for point clouds. In this chapter, we advocate embedding point clouds in Wasserstein space because of its nature of flexibility. The main motive of this study is to examine the advantages of Wasserstein space over widely used Euclidean space for point cloud data. Recent studies show that many spaces can be embedded into Wasserstein space with low distortion [13], this reflects how flexible Wasserstein spaces are. Recently, [7] tries to mimic Wasserstein distance in Euclidean space for image embeddings to build efficient methods along with availing the flexibility of Wasserstein space. There are also a few works that utilize the advantage of Wasserstein space for learning point cloud embeddings, [19], motivated by [7], proposes a method to approximate Wasserstein distance by Euclidean norm between two point cloud embeddings. Since Euclidean space is known for its limited ability, finding *isometric low-distortion* point cloud embeddings is tough. Another work by [23] presents how Optimal Transport (OT) based distances for point cloud reconstruction affect the quality of learnt embeddings. However, this method utilizes OT based distances only as a reconstruction loss, which is not enough to learn *complex shapes* and fails to capture *fine details* of point clouds.

Motivated by the aforementioned limitations and inspired by [13], in this work, we advocate for mapping point cloud as a *discrete distribution* in Wasserstein space. We explain our method in detail next.

4.2 Metric Learning using Optimal Transport

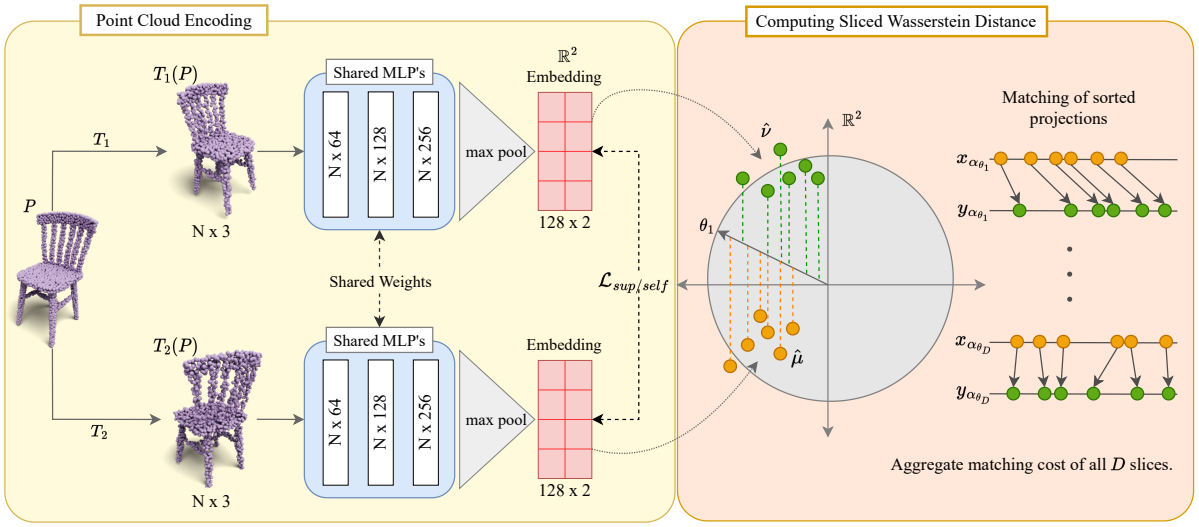


Figure 4.2: **Overview of our proposed method.** The two main parts are, point cloud encoding as discrete distribution (left) and computation of Sliced Wasserstein distance (right). The ground metric space is $\mathbb{R}^2$. $T_1(P)$ and $T_2(P)$ are two instances of P after random transformations.

In this section, we discuss our method of computing Wasserstein embeddings for point clouds in a contrastive learning setup as shown in Figure 4.2. Leveraging the idea of contrasting point clouds against each other, we intend to learn *common and distinctive features* between same and different distributions, respectively. It can be trained in either *supervised* or *self-supervised* manner and can be used as a pre-training methodology for any downstream task. We use *Sliced Wasserstein (SW) distance* which is a low-cost approximation of Wasserstein distance due to its high computational complexity. The goal is to represent samples from same class closer than the samples from different classes in the embedding space (larger inter-cluster and smaller intra-cluster distance). Here, the choice of embedding space plays a key role in desirable performance, as individual metric spaces can embed data differently and represent different types of semantic structure.

Let $O = \{(P_m, l_m)\};\ m = 1 \dots M$ be a collection of point clouds $P_m = \{p_i\};\ i = 1 \dots N_m$, where, $p_i \in \mathbb{R}^3$ with their corresponding class labels $l_m \in L$, where $L = \{1, \dots C\}$ is a set of class labels. Each point cloud P_m contains N_m number of points defined by 3D space points in x, y and z direction. For defining the batch-wise contrastive loss, we first randomly draw K samples from the collection O, that form a batch $B = \{(P_m, l_m)_k\};\ k = 1 \dots K$. For every point cloud $P_m \in B$, we apply fixed set of random transformations T_1 and T_2 to get two instances of P_m (as shown in Figure 4.2), giving an augmented batch $B' = \{(P'_m, l_m)_{k'}\};\ k' = 1 \dots 2K$. The augmented batch is twice the size of the original batch. The point clouds P'_m indexed at k' and $k' + 1$ are augmented versions

of the point cloud P_m indexed at k. As these are augmented versions of $P_{m[k]}$, their class labels are $l_{m[k']} = l_{m[k'+1]} = l_{m[k]}$.

The input to the encoder is an augmented batch B', from which all P'_m needs to be mapped to the embedding space depending on its geometric features and appearance, with samples having same class label being closer. The encoder represents function $f : \mathbb{R}^{N_m \times 3} \rightarrow \mathcal{W}(\mathcal{X})$, that maps a point cloud P'_m to the Wasserstein space $\mathcal{W}(\mathcal{X})$, with W_p being the distance metric on $\mathcal{W}(\mathcal{X})$ and $\mathcal{X}$ being the ground metric space. We choose $\mathbb{R}^2$, $\mathbb{R}^4$ and $\mathbb{R}^8$ to be our ground metric spaces, in which the corresponding embedding z'_m of P'_m is represented as discrete distribution $\{\frac{1}{S} \cdot x_i\}; i = 1 \dots S$ supported by $x_i \in \mathcal{X}$ with a total of S support points, all with uniform probability mass $\frac{1}{S}$ for simplicity. In our implementation, we obtain z'_m by reshaping the encoder's output, to obtain the discrete distribution for different ground metric spaces.

Generally, the computation for exact solution of W_p is costly. To make the computation of optimal transport more tractable, we replace the distance metric W_p on Wasserstein space $\mathcal{W}(\mathcal{X})$ by the Sliced Wasserstein distance metric SW_p. SW_p is a low-cost approximation of Wasserstein distance with computational complexity being $\mathcal{O}(S \log S)$. For all our experiments, we set the value of $p = 2$ and number of slices $D = 300$.

4.2.1 Supervised Contrastive Loss

In the supervised setting, for any $P'_m \in B'$ indexed at k' with corresponding label $l_{m[k']}$, the positive set is defined as $A = \{P'_m \in B' : P'_m = l_{m[k']}\}$. We define our supervised contrastive loss for learning point cloud Wasserstein embeddings as:

$$\mathcal{L}_{sup} = -\sum_{i=1}^{2K} log \left(\sum_{\substack{j \in A \\ j \neq i}} \frac{exp(-SW_2^2(z_i, z_j))}{\sum_{t \neq i} exp(-SW_2^2(z_i, z_t))} \right) \tag{4.1}$$

The loss tries to minimize the Sliced Wasserstein distance between the embeddings represented as discrete distribution of an anchor and all the samples having the same class in the augmented batch. This can also be easily converted to a self-supervised version by making necessary modifications.

4.2.2 Self-Supervised Contrastive Loss

Contrary to the supervised setting, in self-supervised setting, the class label of point clouds cannot be used in any way to train the encoder. Here, the positive set of any $P'_m \in B'$ contains only the other augmentation of P'_m. If $i \in \{1 \dots 2K\}$ be the index of any $P'_m \in B'$, then, let $j(i)$ be the index of its other augmented sample. We define our self-supervised loss for learning point cloud Wasserstein embeddings as:

$$\mathcal{L}_{self} = -\sum_{i=1}^{2K} log \left(\frac{exp(-SW_2^2(z_i, z_{j(i)}))}{\sum_{t \neq i} exp(-SW_2^2(z_i, z_t))} \right) \tag{4.2}$$

Here, only the Sliced Wasserstein distance between embeddings of an anchor and its augmented sample is minimized. Other than the augmented sample, the samples having the same class in the augmented batch are treated as negatives, which might hinder the overall optimization process depending on the batch size.

4.3 Experimental Setup

In this section, we explain the experimental setup for pre-training, all the used datasets, baselines that we compare with, and the used hyperparameters.

4.3.1 Datasets

We use ModelNet10 (MN10) and ModelNet40 (MN40) [41] to perform experiments on classification. MN40 consists of 12311 CAD models with a total of 40 categories, where 9843 objects are used for training and 2468 for testing. We use the data provided by [29], from which we randomly sample 2048 points for each point cloud. MN10 is a subset of MN40 dataset having 10 categories. To evaluate how the learned embeddings perform on real-world data, we also conduct experiments on ScanObjectNN (SO) [36]. It contains object scans with partial occlusions and background, making it a challenging dataset. Here, an object which is represented with 2048 points can also have points that belong to the background. We do not use the mask label that indicates whether a point belongs to the foreground or background in any of our experiments. This interpretation is completely left to the network to decide and choose which points matter the most in order to classify a particular object. It has 2304 objects for training and 567 for testing from 15 categories. For part segmentation, we use ShapeNetPart (SN) [43] that consists of 16681 point clouds from 16 categories and 50 part categories in total. As a pre-processing step, we always normalize the coordinates of the point clouds and scale the object to bring it inside a unit sphere. Note that, the point clouds are not explicitly aligned across all datasets.

4.3.2 Architecture and Pre-training

We use a 3-layer MLP network as our encoder for all the classification and segmentation tasks. The number of neurons in each layer are $(64, 128, 256)$ with batch-norm and ReLU activation after every layer. Note that, the last layer's output is not passed through the activation function. After the max-pooling operation, the encoder gives a 256-dimensional vector, which is then converted into a discrete distribution z'_m depending on the selected ground metric space $(\mathbb{R}^2, \mathbb{R}^4, \mathbb{R}^8)$. For example, if the selected ground metric space is $\mathbb{R}^2$, then the 256-dimensional embedding vector is reshaped into $(256/2) \times 2$, giving us 128 support points in 2 dimensions. As we use a uniform distribution, each of the points have the same weightage of $(1/128)$, all effectively making it a discrete distribution in $\mathbb{R}^2$. For interpolation, we use the encoder and decoder proposed by FoldingNet [42].

In order to perform any downstream task on a particular dataset, the encoder is first pre-trained on the dataset using the contrastive loss explained in Section 4.2 with different distance metrics, followed by testing and evaluation of the desired task. Throughout the experiments, we refer the encoder trained using our method as CL+SW_2. We train CL+SW_2 with ground metric dimensions 2, 4, and 8 and report the results of the best-performing network. In the case of Euclidean distance metrics, the encoder function $f : \mathbb{R}^{N_m \times 3} \to \mathbb{R}^d$ maps a point cloud to d-dimensional space. We then apply L^2-distance or cosine similarity as distance measures on these d-dimensional embeddings. For the cosine similarity metric, we remove the negative sign in the numerator for Eqs. 4.1 and 4.2. Also, note that when training the encoder with cosine similarity, the embeddings are normalized. For Euclidean distances, we use the 256-dimensional vector straight away, and for the Wasserstein metric, we use the discrete representation. For the transformations required in contrastive loss, intended towards forming augmented instances, we sequentially compose random scaling, rotation, and point jittering. The scaling parameter is randomly selected from a range of 0.8 to 1.25. For rotation, we only rotate the object about its Y-axis and choose a rotation angle between 0° to 360°. For points jittering, we first draw noise from a normal distribution having 0 mean and standard deviation as 0.01. This noise is then added to each point individually, introducing surface distortion. We compose these transformations by selecting two random input parameters for each of them, resulting in T_1 and T_2.

4.3.3 Baselines

For computing Euclidean embeddings, we consider L^2-distance and Cosine similarity as distance measures. For these Euclidean metric baselines, we train the encoder using our loss (Eqs. 4.1, 4.2) by replacing $SW_2^2(\cdot, \cdot)$ with the respective metrics. We also consider WPCE [19] and SSW-AE [23] as our baselines as they are recent techniques that use Wasserstein metric for learning point cloud embeddings and are the closest comparable works to our approach. WPCE embeds Wasserstein space into Euclidean space using a Siamese network. The network is trained in such a way that the Euclidean distance mimics the Wasserstein distance between two point clouds. SSW-AE proposed to use SW distance and its variants (max SW and adaptive SW) for reconstruction to learn point cloud embeddings. It examines the effect of using different reconstruction metrics and losses for training an auto-encoder architecture on the learnt embeddings. For a fair comparison, all reported results (apart from interpolation) of our method and baselines are using the same encoder architecture.

4.3.4 Hyperparameters

We perform all our experiments on NVIDIA RTX-2080Ti GPUs using the PyTorch framework for implementing our models. We set the batch size to 16, and the learning rate as 0.001 with a step learning rate scheduler, where the learning rate is scaled by 0.7 after every 20 epoch. We use the Adam optimizer and set weight decay to 0.0001 and momentum to 0.9.

4.4 Downstream Tasks

To demonstrate the representation power of the learned Wasserstein embeddings compared to Euclidean embeddings and other baselines, in this section, we present qualitative and quantitative evaluations on multiple tasks: point cloud classification, transfer learning, point cloud segmentation, and point cloud interpolation with both supervised and self-supervised pre-training settings.

Table 4.1: Results of 3D object classification with supervised and self-supervised pre-training on ModelNet10, ModelNet40 and ScanObjectNN (referred as ScanObject) datasets. WPCE and SSW-AE are unsupervised methods and cannot be evaluated for supervised pre-training (represented by "-"). Bold represents the best result.

Method	Supervised Pre-training			Self-Supervised Pre-training		
	ModelNet10	ModelNet40	ScanObject	ModelNet10	ModelNet40	ScanObject
CL+L^2	90.85	84.64	62.82	90.63	84.72	63.51
CL+Cosine	85.90	70.42	56.11	85.90	72.64	56.45
WPCE	-	-	-	89.97	78.84	52.83
SSW-AE	-	-	-	88.88	76.86	51.29
CL+SW_2 (Ours)	**91.85**	**85.90**	**63.16**	**91.96**	**85.73**	**63.85**

4.4.1 3D Object Classification

We extract point cloud embeddings from a pre-trained encoder and use a linear SVM as our classifier for simplicity. Particularly, we fit a linear SVM classifier on the embeddings acquired by an encoder on the train split and report the overall classification accuracy on the test split. In Figure 4.1, we can see that features captured by Wasserstein embeddings effectively summarize the overall object geometry compared to embeddings learned in Euclidean space. This property also reflects in the classification performance shown in Table 4.1. We can observe that for both supervised and self-supervised settings, the classification accuracy with embeddings extracted by the encoder trained with CL+SW_2 is higher than that of CL+L^2 and CL+Cosine. Thus, compared to Euclidean space, the performance of SW_2 is consistently better on all the datasets, which implies that embeddings learnt in Wasserstein space can increase classification accuracy.

We also show that our method is more effective compared to WPCE and SSW-AE. This improvement can be explained by the difference in the approach of extracting Wasserstein embeddings, where in, our methodology introduces usage of OT metric to directly operate in embedding space endowed by

contrastive learning. It helps in learning better representations by exploiting the similarities between distributions along with utilizing the flexibility of the target Wasserstein space.

4.4.2 Transfer Learning

We examine the generalizing ability of the embeddings acquired by encoders trained with different distance metrics to unseen classes by performing transfer learning for point cloud classification. We follow the same process as explained in Section 4.4.1 for reporting the overall classification accuracy. The quantitative comparisons of transfer learning are shown in Table 4.2. We perform evaluation in two transfer learning settings, MN10 to MN40 and SN to MN40. Here, the encoder is pre-trained on MN10 and SN, followed by evaluation on MN40. In both settings, the model generalizes to new unseen classes by wielding the knowledge of geometry learned during training. We can see that CL+SW_2 consistently performs better than other distance measures and methods in both the transfer learning settings with and without supervision. Results imply that Wasserstein embeddings are better in transferring the knowledge of capturing geometry for yielding good classification performance.

Table 4.2: Results of 3D object classification with supervised and self-supervised pre-training for *transfer learning* setup. WPCE and SSW-AE are unsupervised methods and cannot be evaluated for supervised pre-training (represented by "-"). Bold represents the best result.

Method	Supervised Pre-training		Self-Supervised Pre-training	
	MN10 to MN40	SN to MN40	MN10 to MN40	SN to MN40
CL+L^2	85.37	85.81	84.27	83.83
CL+Cosine	74.51	69.12	75.32	71.47
WPCE	-	-	77.51	78.03
SSW-AE	-	-	76.05	76.66
CL+SW_2 (Ours)	**86.18**	**86.18**	**85.61**	**85.77**

4.4.3 3D Object Part Segmentation

We train a 3-layer MLP network to predict a class label for all points in a point cloud, where the input to this network is the embedding provided by a pre-trained encoder. The number of neurons in each layer are $(256, 128, 50)$, with batch-norm and ReLU after each layer except the last one. In particular, part segmentation requires a fine-grained understanding of the local geometry of the objects. Along with the global embedding of the point cloud, per-point embeddings acquired before max-pooling are

stacked together and passed to the segmentation network. Note that only the segmentation network weights are optimized using the standard cross-entropy loss, and the encoder's weights are frozen. We evaluate the performance using mIoU metric. For mIoU of each class, the IoUs of all parts from that class are averaged. Instance average mIoU is calculated by taking the mean of IoUs for all the instances. The comparison of average instance mIoU and per class average mIoU for both supervised and self-supervised learning settings are shown in Table 4.3. We can see that our results outperform other distance measures and methods, implying that Wasserstein embeddings are able to capture better fine-grain local information required for the task.

Table 4.3: Results of part segmentation on ShapeNetPart dataset with supervised (Sup) and self-supervised (S-Sup) pre-training. Bold represents the best results, and underline represents the second best.

Method	Pre-Train	mIoU	aero	bag	cup	car	chair	ear phone	guitar	knife	lamp	laptop	motor	mug	pistol	rocket	skate board	table
CL+L^2	Sup	77.61	73.54	62.76	72.41	64.75	82.42	65.41	88.95	83.01	75.83	93.52	43.03	83.25	74.87	46.01	62.39	77.54
	S-Sup	78.94	75.40	62.74	72.67	67.73	84.73	68.02	89.19	**83.35**	76.98	94.48	43.15	84.19	75.11	49.60	67.81	77.91
CL+Cosine	Sup	74.75	67.80	62.13	**78.66**	66.26	80.00	61.14	86.47	79.34	74.25	92.24	48.70	84.91	70.82	45.63	63.97	73.12
	S-Sup	78.49	72.51	68.09	71.44	68.35	84.47	63.38	88.69	80.30	75.94	94.45	48.81	88.56	74.08	47.37	69.12	77.90
WPCE	S-Sup	79.92	77.47	**69.06**	74.22	66.59	86.79	66.23	89.30	81.77	75.97	94.60	42.29	**88.71**	74.33	41.05	67.39	79.28
SSW-AE	S-Sup	75.20	68.83	59.61	69.65	64.23	81.98	62.00	86.92	80.27	73.70	92.82	38.46	85.12	68.26	43.97	60.65	73.69
CL+SW_2 (Ours)	Sup	**81.40**	**80.50**	64.69	74.41	**70.97**	**87.34**	69.71	89.34	82.96	77.59	**95.31**	**57.28**	88.03	**77.14**	53.18	**69.60**	79.84
	S-Sup	81.17	78.98	**66.90**	77.98	70.35	86.91	**70.57**	**89.39**	82.85	**77.99**	94.77	56.20	87.18	76.10	**53.98**	69.60	**80.10**

4.4.4 3D Shape Interpolation

We further examine the quality of our learnt space by performing shape interpolation between inter and intra class point cloud instances. The main aim of conducting this task is to examine which learnt space is capable of capturing geometric and structural information needed to generate consistent interpolations of 3D point clouds. One can inspect the quality of latent space by examining the smoothness of an interpolation path or the quality of the in-between generated samples in terms of noise. As interpolation is a synthesis task, we need a decoder network to reconstruct the object given its embedding. For this, we train an encoder-decoder network with our contrastive loss (Eq. 4.1) on the embeddings for the encoder, along with a reconstruction loss for the decoder with 512 as the embedding size. We use the encoder and decoder proposed by FoldingNet [42], which learns to deform a unit sphere and take the shape of a 3D object's surface. We found that optimizing the network for learning discriminative features as well as detailed reconstruction is difficult. As our contrastive loss aims to pull point clouds closer with similar global representations, it becomes difficult to accurately reconstruct the input point cloud without fine-grain characteristic information. A simple way to deal with this issue is to assign

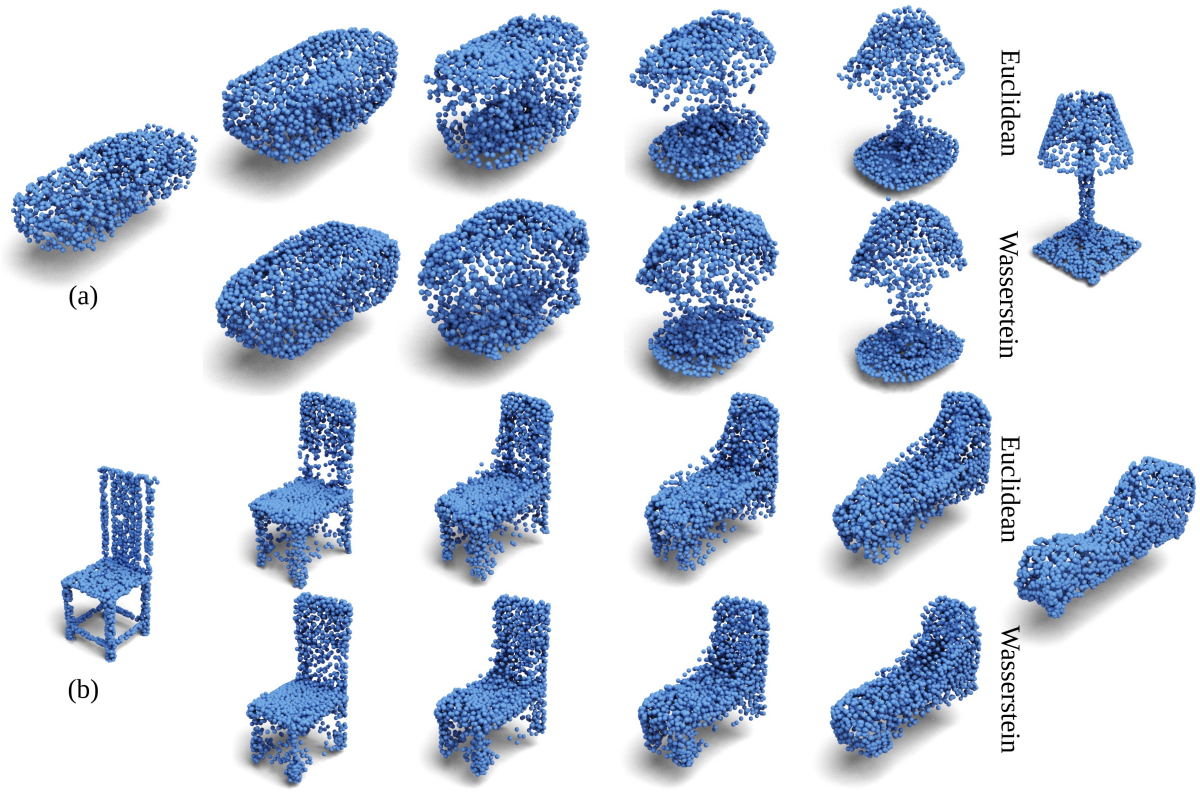


Figure 4.3: **Linear Interpolation** between source and target for two examples (a) Car to Lamp, (b) Chair to Chair from ShapeNet. The left and the right most represent original point clouds. The top row of each example shows results of reconstruction after interpolating two point cloud embeddings in Euclidean space. The bottom row of each example provides interpolation results in Wasserstein space. All rows follow the ratio of 0.8, 0.6, 0.4, and 0.2 (from left to right) with respect to the source.

weightage to the individual loss terms, with the weights summing to 1. In order to train an encoder-decoder network, the total effective loss is defined by taking a weighted sum of our contrastive loss and a reconstruction loss, with weights being 0.2 and 0.8, respectively. We use Chamfer distance as the reconstruction loss. Interpolation results are shown in Figure 4.3. We can see that the interpolations done using Wasserstein embeddings follow a smooth path with relatively less noisy points. For example, in Figure 4.3 (a), we can see that for Euclidean (in step 2 to 3), the interpolated sample suddenly takes the shape of a lamp. Whereas the interpolation path is smoother for Wasserstein. Similar trend can also be observed in Figure 4.3 (b), where for Euclidean, the source chair suddenly transforms (in step 2 to 3) to take the shape of the target chair, whereas in Wasserstein, legs of the chair smoothly morph to become the base of target chair. We evaluate the quality of interpolated samples based on the noise present in them. For each interpolated sample, we compute the noise measure given in Eqn 4.3.

$$\mathcal{L}_{noise} = \sqrt{\frac{1}{N_m}\sum_{i=1}^{N_m}\left\| p_i - \frac{1}{n_e}\sum \mathcal{N}(p_i)\right\|^2} \tag{4.3}$$

where, $\mathcal{N}(p_i)$ gives a set of neighboring points for p_i that has a cardinality of $n_e = 20$. For a given point cloud, the noise measure computes neighborhood variation vectors around every point and aggregates their squared norm to give an overall score of smoothness. A lower noise measure implies that the point cloud has a smooth surface and has less noise. In Table 4.4, we report the noise measure of the interpolated samples shown in Figure 4.3. Most of the interpolated samples from Wasserstein space have lower values of noise measure. We provide more examples in Figure 4.5 and report their noise measure results in Table 4.5.

Table 4.4: Noise measure for interpolation results shown in Figure 4.3. Bold values represent smoother surfaces having less noise. The values are scaled by a factor of 10^3.

Figure	Method	Step 1	Step 2	Step 3	Step 4
4.3 (a)	Euclidean	30.2	35.6	36.7	35.4
	Wasserstein	**29.6**	**32.1**	**32.6**	**31.4**
4.3 (b)	Euclidean	**29.3**	29.9	30.2	28.0
	Wasserstein	29.7	**29.1**	**29.4**	**27.7**

Table 4.5: Noise measure for interpolation results shown in Figure 4.5. Bold values represent smoother surfaces having less noise. The values are scaled by a factor of 10^3.

Figure	Method	Step 1	Step 2	Step 3	Step 4
4.5 (a)	Euclidean	31.5	36.8	**34.6**	**30.5**
	Wasserstein	**30.0**	**34.9**	35.0	31.8
4.5 (b)	Euclidean	29.8	31.8	33.4	**33.4**
	Wasserstein	**29.1**	**30.1**	**33.0**	34.5
4.5 (c)	Euclidean	**25.0**	**24.3**	**23.3**	**22.3**
	Wasserstein	25.3	24.4	**23.3**	22.5
4.5 (d)	Euclidean	24.9	26.2	26.0	**25.0**
	Wasserstein	**23.2**	**24.7**	**25.1**	25.3

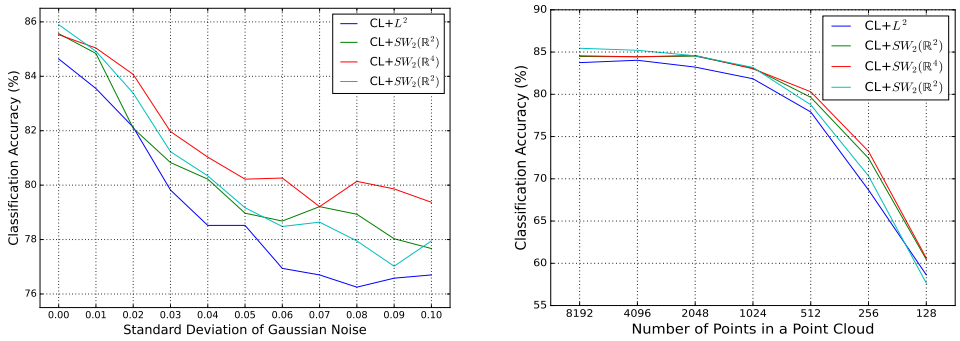


Figure 4.4: **Ablation Study:** Classification for noise model with Gaussian Noise (top) and points removal using random sampling (bottom) for our method CL+SW_2 and baseline CL+L^2.

4.5 Ablation Study: Classification Task

We perform point perturbation and point density variation to test their effects on the encoders pretrained with different distance metrics and report the classification accuracy on Modelnet40 as shown in Figure 4.4. For the point perturbation test, we add Gaussian noise to input point clouds, with a standard deviation of noise varying from 0.01 to 0.1. We can observe that for all noise levels, even with severe distortion, CL+SW_2 performs well than that of CL+L^2. This implies that discrete representation learnt in Wasserstein space is less prone to performance degradation due to noise in inputs. Further, for varying density test, we randomly sample 8192, 4096, 2048, 1024, 512, 256, and 128 points from input point clouds and perform evaluation on them. We can observe that CL+SW_2 consistently does better than CL+L^2. This shows Wasserstein embeddings are robust towards missing points in the input point cloud.

4.6 Explainability

We investigate what makes Wasserstein embeddings perform better, as shown in the downstream tasks. We visualize and compare the features captured by Wasserstein embeddings and Euclidean embeddings in Figure 4.1. These features are called critical points, as shown by [28]. The embedding of a point cloud is completely determined by these subset of points. The embedding for a point cloud would be the same, as long as the set of critical points is unchanged. For a given point cloud, the critical points are those 3D points that contribute to the global embeddings after the max pooling layer. This implies that the number of critical points cannot be greater than that of the embedding size. The selection of critical points is extremely important, as they solely decide the embedding of a point cloud. This makes it clear that for good quality embeddings, critical points should best describe the given point cloud. In Figure 4.1, we can see that the network intelligently tries to summarize the point cloud by choosing boundary points as the critical points. Our Wasserstein embeddings are able to capture the full skeleton structure of the given point cloud, whereas critical points captured by Euclidean embeddings are com-

paratively poor with uneven distribution and missing parts. Thus, we can say that Wasserstein spaces are indeed better at preserving and capturing geometric structures amenable to the optimization task. We provide more example in Figure 4.6, 4.7.

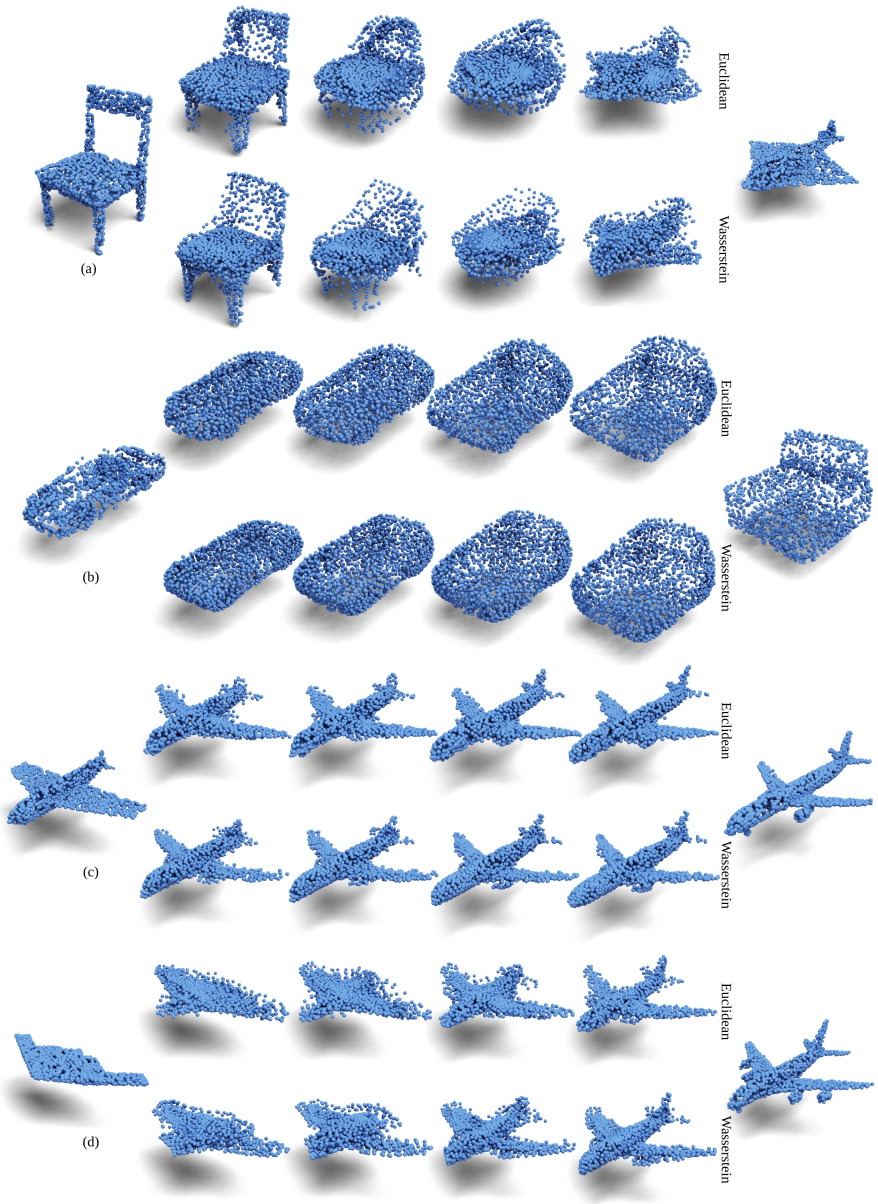


Figure 4.5: Comparision of interpolation from source (leftmost) to target (rightmost) samples between Euclidean embeddings and Wasserstein embeddings. For each sample, top row shows results from reconstruction after interpolating two point cloud embeddings in Euclidean space. Likewise, bottom row shows results obtained by Wasserstein embeddings. Weight ratio $(0.8, 0.6, 0.4, 0.2)$ (from left to right) with respect to the source.

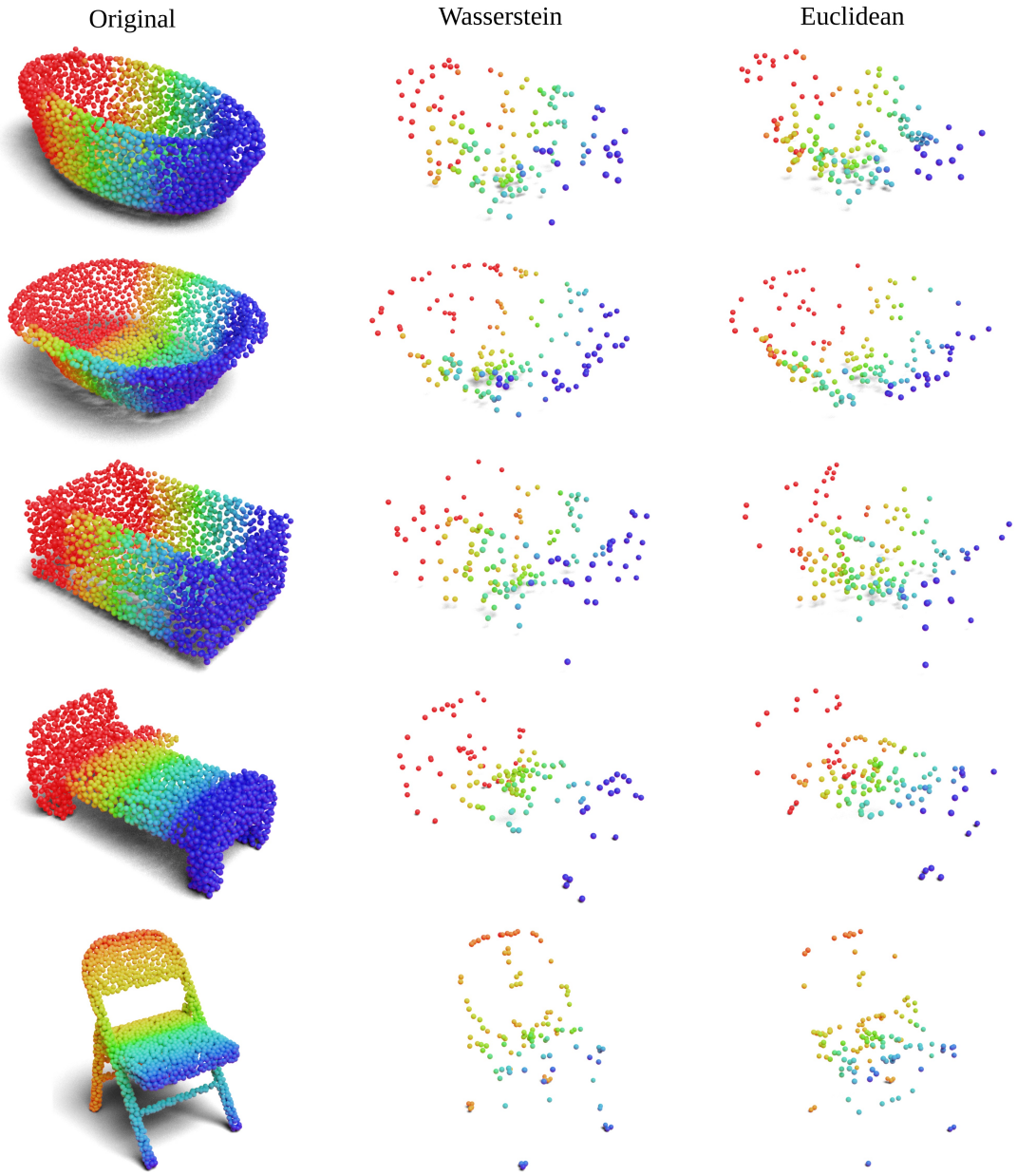


Figure 4.6: Comparison of important points given by network trained with Wasserstein and Euclidean metric. First column shows the original point cloud, second column shows the critical points set captured by Wasserstein space, and third column shows the same for Euclidean space.

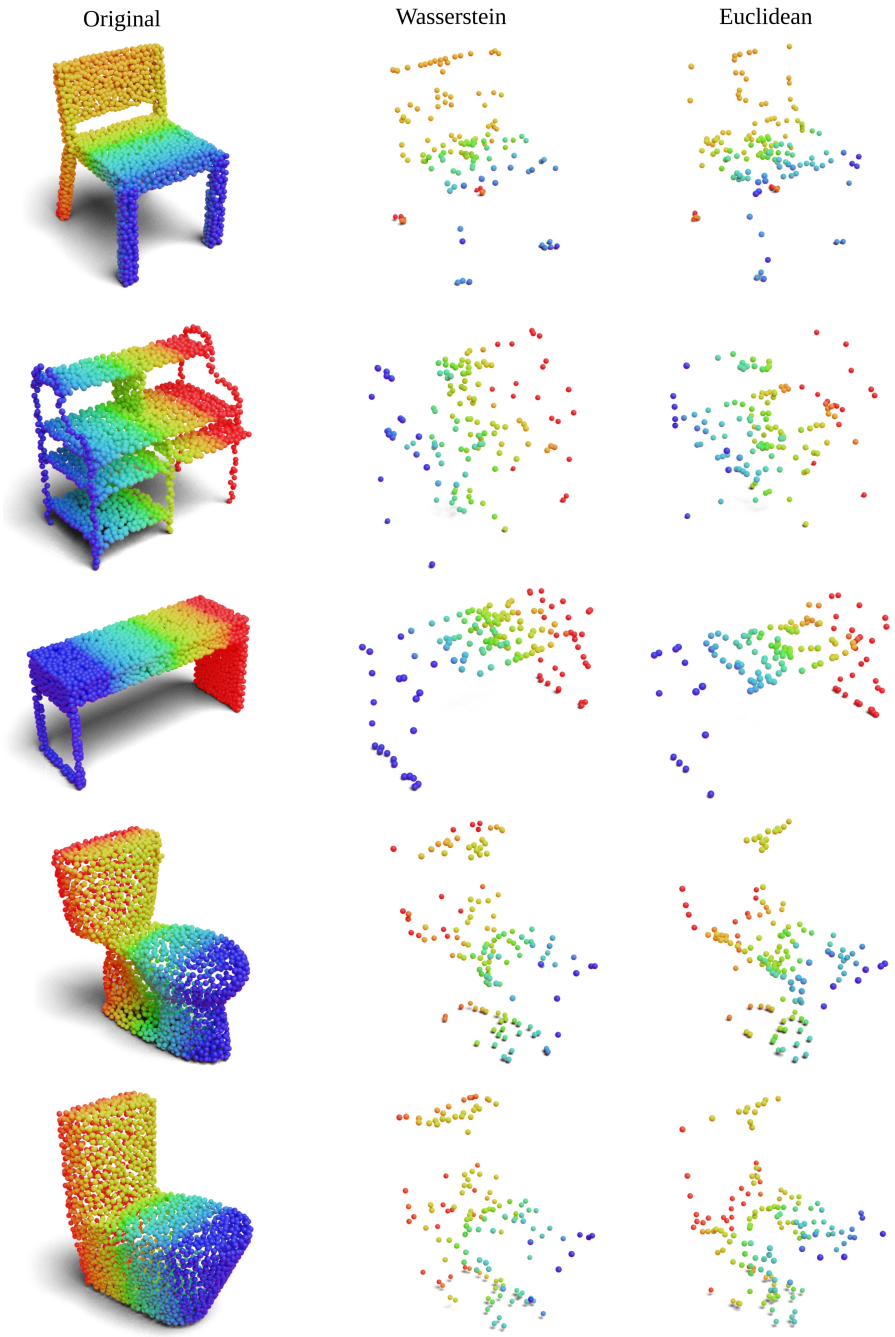


Figure 4.7: Comparison of important points given by network trained with Wasserstein and Euclidean metric. First column shows the original point cloud, second column shows the critical points set captured by Wasserstein space, and third column shows the same for Euclidean space.

4.7 Summary

In this chapter, we presented a contrastive learning method to learn Wasserstein embeddings for 3D point clouds. We proposed to represent point clouds as discrete probability distributions in the Wasserstein space. Our proposed method can be used as a pre-trained model in supervised and self-supervised settings for any downstream task. Empirically, we found that representations learnt using our pre-training of contrastive learning with Sliced Wasserstein distance captured the structure and underlying geometry better than standard Euclidean embeddings. With improved embeddings, our method outperformed all the existing methods, including our baseline with L^2 norm and Cosine similarity for all the downstream tasks (classification, segmentation, transfer learning, interpolation). We also show an interesting study of our self-explainable method by capturing critical points of point clouds better than embeddings in Euclidean space. As a future direction, it would be interesting to study and explore how to inject the properties of Wasserstein space as shown in our method in bigger models like transformers and also explore how it would help more complex tasks like 3D reconstruction and 3D generation.

Chapter 5

Conclusion

In this thesis, we proposed two methods for 3D representation learning endowed by Optimal Transport. Particularly, we looked at the problem of point cloud domain adaptation and learning Wasserstein embeddings for point clouds. Here is a summary of the thesis.

In Chapter 2, we give a brief introduction to Optimal Transport. We look at how this problem of transportation evolved from practical use cases to more advance research use cases. We then describe multiple distance metrics derived from Optimal Transport, them being, p-Wasserstein distance and Sliced-Wasserstein distance that are crucial to understanding our proposed methods in upcoming chapters.

In Chapter 3, we first motivate the need for optimal transport and multimodal inputs towards solving the task of point cloud domain adaptation. We then explain our method COT, where the main idea is to synergize the working of contrastive learning and optimal transport for explicit domain alignment. Finally, we show exhaustive experiments on multiple datasets and achieve SoTA results on one of them. Our ablation studies also validates the importance of individual components used in our method. We also perform an interesting experiment of visualizing the decision boundaries with and without our method in action, both for early and final epochs.

In Chapter 4, we first discuss the limitations of Euclidean spaces and motivate the need to go beyond them to learn effective point cloud embeddings. We then explain our method endowed by optimal transport and contrastive learning to learn Wasserstein embeddings. To show the effectiveness of our method, we perform multiple downstream tasks in both supervised and self-supervised settings and compare with multiple baselines that use Euclidean as well as Wasserstein metric. Finally, we also visualize and compare the features learned by the model trained with Euclidean and Wasserstein metrics.

To conclude, in this thesis, we propose two novel methods endowed by optimal transport for 3D point cloud representation learning for multiple tasks in multiple settings, them being: unsupervised domain adaptation, supervised and self-supervised settings. We hope this work will motivate the community to carry out research on using the power of optimal transport for 3D understanding or even 3D generative tasks.

Related Publications

- **Synergizing Contrastive Learning and Optimal Transport for 3D Point Cloud Domain Adaptation**, *Siddharth Katageri**, Arkadipta De*, Chaitanya Devaguptapu*, VSSV Prasad, Charu Sharma, Manohar Kaul.
  In IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) 2024, **Oral**.

- **Metric Learning for 3D Point Clouds Using Optimal Transport**, *Siddharth Katageri*, Srinjay Sarkar, Charu Sharma.
  In IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW) 2024.

Bibliography

[1] I. Achituve, H. Maron, and G. Chechik. Self-supervised learning for domain adaptation on point clouds. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 123–133, 2021.

[2] M. Afham, I. Dissanayake, D. Dissanayake, A. Dharmasiri, K. Thilakarathna, and R. Rodrigo. Crosspoint: Self-supervised cross-modal contrastive learning for 3d point cloud understanding. *Ieee/cvf Conference On Computer Vision And Pattern Recognition (cvpr)*, 2022.

[3] A. Agrawal, J. Lu, S. Antol, M. Mitchell, C. L. Zitnick, D. Parikh, and D. Batra. Vqa: Visual question answering. *Int. J. Comput. Vision*, 123(1):4–31, may 2017.

[4] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv: Arxiv-1512.03012*, 2015.

[5] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning (ICML)*, pages 1597–1607. PMLR, 2020.

[6] Y. Chen, V. T. Hu, E. Gavves, T. Mensink, P. Mettes, P. Yang, and C. G. M. Snoek. Pointmixup: Augmentation for point clouds. In A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, editors, *Computer Vision – ECCV 2020*, pages 330–345, Cham, 2020. Springer International Publishing.

[7] N. Courty, R. Flamary, and M. Ducoffe. Learning wasserstein embeddings. In *International Conference on Learning Representations (ICLR)*, pages 1–13, 2018.

[8] M. Cuturi and D. Avis. Ground metric learning. *J. Mach. Learn. Res.*, 15(1):533–564, jan 2014.

[9] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. *Ieee Conference On Computer Vision And Pattern Recognition (cvpr)*, 2017.

[10] B. Damodaran, B. Kellenberger, R. Flamary, D. Tuia, and N. Courty. Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation. *ECCV*, 2018.

[11] H.-S. Fang, C. Wang, M. Gou, and C. Lu. Graspnet-1billion: A large-scale benchmark for general object grasping. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11441–11450, 2020.

[12] R. Flamary, N. Courty, A. Gramfort, M. Z. Alaya, A. Boisbunon, S. Chambon, L. Chapel, A. Corenflos, K. Fatras, N. Fournier, L. Gautheron, N. T. Gayraud, H. Janati, A. Rakotomamonjy, I. Redko, A. Rolet, A. Schutz, V. Seguy, D. J. Sutherland, R. Tavenard, A. Tong, and T. Vayer. Pot: Python optimal transport. *J. Mach. Learn. Res.*, 22(1), jul 2022.

[13] C. Frogner, F. Mirzazadeh, and J. Solomon. Learning entropic wasserstein embeddings. In *International Conference on Learning Representations (ICLR)*, 2019.

[14] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. March, and V. Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35, 2016.

[15] A. Hamdi, S. Giancola, and B. Ghanem. Mvtn: Multi-view transformation network for 3d shape recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1–11, October 2021.

[16] R. Hanocka, A. Hertz, N. Fish, R. Giryes, S. Fleishman, and D. Cohen-Or. Meshcnn: A network with an edge. *ACM Trans. Graph.*, 38(4), jul 2019.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *Ieee Conference On Computer Vision And Pattern Recognition (cvpr)*, 2015.

[18] K. Kafle and C. Kanan. Visual question answering: Datasets, algorithms, and future challenges. *Computer Vision and Image Understanding*, 163:3–20, 2017. Language in Vision.

[19] K. Kawano, S. Koide, and T. Kutsuna. Learning wasserstein isometric embedding for point clouds. In *International Conference on 3D Vision (3DV)*, pages 473–482, 2020.

[20] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan. Supervised contrastive learning. *Advances in Neural Information Processing Systems (NeurIPS)*, pages 18661–18673, 2020.

[21] R. Mahjourian, M. Wicke, and A. Angelova. Unsupervised learning of depth and ego-motion from monocular video using 3D geometric constraints. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 5667–5675, 2018.

[22] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928, 2015.

[23] T. Nguyen, Q.-H. Pham, T. Le, T. Pham, N. Ho, and B.-S. Hua. Point-set distances for learning representations of 3D point clouds. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[24] M. Nickel and D. Kiela. Poincaré embeddings for learning hierarchical representations. In *International Conference on Neural Information Processing Systems (NeurIPS)*, page 6341–6350, 2017.

[25] M. Nickel and D. Kiela. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *International Conference on Machine Learning (ICML)*, pages 3776–3785. PMLR, 2018.

[26] L. Pan, Ladický, and M. Pollefeys. Compression and completion of animated point clouds using topological properties of the manifold. In *International Conference on 3D Vision (3DV)*, pages 734–742, 2020.

[27] G. Peyré and M. Cuturi. Computational optimal transport. *Foundations and Trends in Machine Learning*, pages 355–602, 2019.

[28] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[29] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *International Conference on Neural Information Processing Systems (NeurIPS)*, page 5105–5114, 2017.

[30] C. Qin, H. You, L. Wang, C.-C. J. Kuo, and Y. Fu. Pointdan: A multi-scale 3d domain adaption network for point cloud representation. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[31] S. A. Rose. Infants' transfer of response between 2-dimensional and 3-dimensional stimuli. *Child Development*, 1977.

[32] J. Sauder and B. Sievers. Self-supervised deep learning on point clouds by reconstructing space. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[33] N. Sharp, S. Attaiki, K. Crane, and M. Ovsjanikov. Diffusionnet: Discretization agnostic learning on surfaces. *ACM Trans. Graph.*, 41(3), mar 2022.

[34] Y. Shen, Y. Yang, M. Yan, H. Wang, Y. Zheng, and L. J. Guibas. Domain adaptation on point clouds via geometry-aware implicits. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7223–7232, June 2022.

[35] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proc. ICCV*, 2015.

[36] M. A. Uy, Q.-H. Pham, B.-S. Hua, D. T. Nguyen, and S.-K. Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *International Conference on Computer Vision (ICCV)*, 2019.

[37] Villani. *Topics in Optimal Transportation*. American Mathematical Society, 2003.

[38] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph CNN for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 2019.

[39] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Trans. Graph.*, 38(5), oct 2019.

[40] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. *Ieee Conference On Computer Vision And Pattern Recognition (cvpr)*, 2014.

[41] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015.

[42] Y. Yang, C. Feng, Y. Shen, and D. Tian. FoldingNet: Point cloud auto-encoder via deep grid deformation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 206–215, 2018.

[43] L. Yi, V. G. Kim, D. Ceylan, I.-C. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, and L. Guibas. A scalable active framework for region annotation in 3D shape collections. *ACM Transactions on Graphics (TOG)*, 2016.

[44] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

[45] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *IEEE international conference on robotics and automation (ICRA)*, pages 3357–3364, 2017.

[46] L. Zou, H. Tang, K. Chen, and K. Jia. Geometry-aware self-training for unsupervised domain adaptation on object point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6403–6412, 2021.