POLITECNICO DI TORINO

Master degree course in Computer Engineering

Master Degree Thesis

Self-Supervised Generalization on Real-World 3D Point Clouds

Supervisor
prof. Tatiana Tommasi

Co-Supervisors:
prof. Barbara Caputo
doct. Davide Boscaini

Candidate
Antonio Alliegro

Academic year 2018-2019

Self-Supervised Generalization on Real-World 3D Point Clouds Master thesis. Politecnico di Torino, Turin.

December 2019.

The work in this thesis has been submitted to the Conference on Computer Vision and Pattern Recognition (CVPR) 2020.

Acknowledgements

Vorrei ringraziare i miei genitori, le mie sorelle Laura ed Elisa, i miei zii, i miei nonni e la mia famiglia tutta la quale mi ha sempre supportato in ogni mia decisione ed incoraggiato a raggiungere i miei obiettivi. Un ringraziamento speciale va a mia madre e mio padre. Lei mi è sempre stata vicino, insegnandomi a dare maggior valore alle piccole vittorie collezionate in questo iter universitario e non. A mio padre, invece, fra le tante cose devo la mia dedizione al lavoro, non posso negare di essermi più volte ispirato a lui ed averlo preso ad esempio, specie nelle nottate passate al computer per 'far funzionare le cose'.
Questo mio ultimo anno da studente a Torino è stato pieno di cambiamenti, fortuna vuole che abbia trovato una seconda famiglia in Pia, Enzo, Eleonora e Gianmaria. Ringrazio Pia ed Enzo, in particolare, per avermi accolto in casa loro con grande gentilezza e, spero, poco disturbo. Vi ringrazio infinitamente per tutto quello che avete fatto per me, grazie per tutti i momenti di condivisione che hanno reso questo periodo meno stressante e più gioioso.
Ringrazio la Professoressa Barbara Caputo per avermi accolto nel suo gruppo di ricerca ed aver creduto nel progetto a cui ho lavorato nonché di avermi fornito tutti i mezzi e le risorse necessarie per lavorare al meglio. L'esperienza al VANDAL group mi ha fortemente arricchito, mi ha permesso di conoscere un team di abili ricercatori che hanno, ognuno in modo diverso, contribuito alla mia formazione.
Vorrei dedicare un ringraziamento speciale alla Professoressa Tatiana Tommasi e al Dott. Davide Boscaini che hanno lavorato attivamente con me a questo progetto di tesi. Quando ripenserò al periodo della tesi ricorderò le infinite chiamate con Davide che, sebbene a distanza, anche nei momenti meno ovvi è sempre riuscito ad indicarmi la direzione giusta da seguire. Entrambi siete stati fondamentali per la buona riuscita di questo lavoro, sono profondamente grato per l'opportunità di lavorare al vostro fianco. Grazie a voi ora ho un'idea concreta di cosa vuol dire 'fare ricerca', mi avete profondamente ispirato e convinto a continuare il mio lavoro in questo ambito. Non vedo l'ora di ripartire con il prossimo progetto!
Infine, vorrei ringraziare tutti i miei amici. Il mio cambiare spesso città non mi ha mai permesso di identificare un posto in particolare come 'casa', tuttora quando qualcuno mi chiede 'Da dove vieni?' fatico a rispondere, allo stesso tempo però mi ha permesso di conoscere tante persone. Mi considerò un pò una 'spugna', mi piace prendere il meglio delle persone con cui vengo a contatto e voi, ognuno in modalità e ambiti differenti,

avete contribuito a formare la persona che ora sono. Ormai abbiamo preso tutti strade diverse, viviamo in città e talvolta stati diversi, ma sento di poter ancora contare su tutti voi. In particolare ringrazio l'amico di sempre Nicola, il quale mi ha sempre supportato/sopportato e che per cinque lunghi anni ha subito le mie telefonate del 'giorno prima dell'esame', ora dovresti essere più al sicuro (forse). Ringrazio Gaia, per essere sempre stata quella più matura e genuinamente buona, lo svizzero Guido, il compagno di banco Raffaele, Teresa, Maura ed Amina. Ringrazio Giulio, Luca, Valeria, Renato, Armando, Eze, Laura, Sofi e tutti gli amici argentini, il tempo degli 'Escaladores Seriales' è stato quasi surreale, in assoluto il più felice e spensierato della mia vita, spero di rivedervi presto tutti in qualche posto remoto del mondo con più montagne di Santa Rosa La Pampa, gracias por todo, nos vemos pronto para mate y escalada!

Abstract

Point cloud processing and 3D shape understanding are very challenging tasks for which deep learning techniques have demonstrated a great potential. Still further progresses are essential to allow artificial intelligent agents to interact with the real world, where the amount of annotated data may be limited and integrating new sources of knowledge becomes crucial to support autonomous learning. Here we consider several possible scenarios involving synthetic and real-world point clouds and we propose to combine supervised and self-supervised learning. We introduce a multi-task model that can solve a 3D puzzle while learning the main task of shape classification or part segmentation on the available annotated data. An extensive analysis over several state-of-the-art datasets investigating transfer learning and cross-domain settings shows the effectiveness of our approach.

Contents

List of Figures VIII

List of Tables IX

1 Introduction 1
- 1.1 Contextual Overview . . . 1
- 1.2 A matter of Perception . . . 2
- 1.3 Motivation . . . 3
- 1.4 Contribution . . . 4

2 Related Works 6

3 Background 9
- 3.1 Self-Supervised Learning . . . 9
- 3.2 Multi-Task Learning . . . 10
- 3.3 Domain Shift . . . 11
  - 3.3.1 Transfer Learning . . . 12
  - 3.3.2 Domain Adaptation . . . 13
  - 3.3.3 Domain Generalization . . . 14
- 3.4 A leap into 3D and Real World . . . 15

4 Geometric Deep Learning 17
- 4.1 Euclidean Data . . . 18
- 4.2 Non-Euclidean data . . . 19
- 4.3 Point Clouds . . . 19
- 4.4 Geometric Deep learning Tasks . . . 20
- 4.5 PointNet . . . 22
- 4.6 PointNet++ . . . 24

5 Datasets 26
- 5.1 Synthetic vs Real-World data . . . 26
- 5.2 Modelnet . . . 27

5.3 Shapenet . . . 29
5.4 ShapeNet Part . . . 29
5.5 ScanObject . . . 30

6 The Method 32
6.1 Jigsaw Generalization . . . 32
6.2 Intuition . . . 33
6.3 Method Formalization . . . 34
6.4 Implementation Choices . . . 35

7 Experiments 39
7.1 Preface . . . 39
7.2 Settings . . . 40
7.3 Object Classification Results . . . 41
7.3.1 Single Domain . . . 41
7.3.2 Transfer Learning . . . 42
7.3.3 Limited Annotated Data . . . 43
7.3.4 Domain Adaptation and Generalization . . . 44
7.4 Part Segmentation Results . . . 45
7.4.1 Few Shot and Semi-Supervised Results . . . 46
7.4.2 Single Domain and Transfer Learning on Real World data . . . 47
7.4.3 Domain Generalization and Adaptation . . . 49

8 Conclusions 51

List of Figures

3.1 Multi-Task with Hard Parameter Sharing 11
3.2 DANN: Domain-Adversarial Neural Networks 14

4.1 3D data representations 17
4.2 Point Cloud, example from ModelNet40 20
4.3 Part Segmentation: example for lamp object 22
4.4 PointNet Architecture 24
4.5 PointNet++ Architecture 24

5.1 ModelNet40 example shapes 28
5.2 ScanObject chair perturbations 31

6.1 Our multi-task architecture on top of PN1 35
6.2 Our multi-task architecture on top of PN2 35
6.3 Our multi-task for Object Classification 36
6.4 3d Jigsaw solving: the puzzle gets solved through epochs 38

7.1 Our multi-task for Part Segmentation 46
7.2 Part Segmentation: our method vs baseline in case of only 1% supervision (chairs visualization). 47
7.3 Part Segmentation: only 1% of supervision, ours vs baseline, chair detailed visualization. 48
7.4 Part Segmentation: only 1% of supervision, ours vs baseline (lamps visualization). 48
7.5 Part Segmentation: only 1% of supervision, ours vs baseline, lamp detailed visualization. 48

8.1 Our multi-task for both Obj. Classification and Part Segmentation 51

List of Tables

5.1 ModelNet40 categories distribution 28
5.2 ShapeNetCore categories distribution 30

7.1 Classification scores: Single Domain and TL 43
7.2 Obj. Classification: accuracy scores in case of limited training data 44
7.3 Obj Classification: accuracy scores for Domain Adaptation and Generalization settings, emphasis is placed on mitigating the domain shift between synthetic and real-world domain 45
7.4 Part Segmentation with only 1% of supervision 47
7.5 Part Segmentation: accuracy scores on ScanObject dataset 49
7.6 Part Segmentation: accuracy scores on ScanObject in DA and DG settings 50

Chapter 1

Introduction

1.1 Contextual Overview

We are living interesting times, what machines are able to do nowadays is constantly shifting through new scenarios. A growing-up number of semantic perceptual tasks is getting solved by machines at near-superhuman levels of performance. Although it's rather reckless speaking about machine thinking and acting as humans, in a safer way we refer to algorithms approximating nonlinear processes efficiently. Those generally fall under the name of Deep Learning techniques. At the base of Deep Learning techniques there are Deep Neural Networks, mathematical entities able to learn from examples which enabled the overall field of Artificial Intelligence. AI has achieved a great leap in recent years with terrific results in the area of computer vision. This naturally reflects also in our every-day life with apps able to collect images and automatically classify object, detect faces, recognize places and so on. Still the task of fully understanding our real world remains far-fetched for many reasons ranging from the inherent difficulty of dealing with a three-dimensional space as well as time and domain variations which makes any prediction unstable. Currently those issues are the same that maintain a large gap between having a smartphone in our hands and a robot at home. Embodied intelligent systems need more than 2D-visual perception: 3D-shapes have to be reliably recognized regardless of the environmental constraints and possibly segmented in their functional parts to allow a robot completing a simple task as can be opening a honey jar. Thanks to the rise of powerful computational resources, 3D research is also progressively flourishing together with new ways to collect and describe 3D data. LiDAR scanners and stereo cameras gave rise to massive point cloud datasets possibly spanning even large entities such as an entire city. However, they come with three main drawbacks: point clouds are unstructured, unordered and eager for precise manual annotation due to the many causes of

noise. The first two properties make typical convolutional neural networks (CNN) unsuitable for point clouds. Possible solutions consists in rendering point clouds to multiple 2D views or pass through a separate voxelization step only followed by 3D CNN, but these techniques are either computationally expensive or come with inevitable loss of information with negative effects on the overall recognition or segmentation performance. The third property has initially guided research towards very well lab-controlled and synthetic CAD object datasets where labeling is simpler. However the most recent results on those kind of testbed are witnessing a trend of performance saturation raising the question of how to move forward. All these challenges describe a research area in need of new solutions for effective and efficient deep learning models able to deal with large amount of unsupervised real-world point cloud data.

1.2 A matter of Perception

Despite the impressive results achieved by CNN architectures in the 2D domain (RGB images), research on 2D data also highlighted the need for huge labeled training dataset as ImageNet. Large scale 3D CAD labelled models datasets, as ModelNet [1] and ShapeNet [2], have been released only few years ago (2015), opening the door for more experiments and boosting the research in this area. Large data availability, cheaper and more accurate data acquisition devices, highly parallel GPU computing affordability, all these factors together motivated the push for 3D deep learning, aiming to exploit 3D data representations in end-to-end deep learning algorithms. 3D data directly encode the geometric properties of sensed objects, pushing computer vision beyond the inherent limitations of a 2D representation. For an increasing number of AI application fields, such as object manipulation, autonomous vehicles, car detection *etc.* 2D images does not provide sufficient information to leverage for the task accomplishment.
Intelligent systems, to be such, must be aware of what is around them in terms of dimension, location and orientation; they should also be able to interact with the surrounding elements in an effective and reasonable way. To allow intelligent systems build a robust *comprehensive perception* of what is around, perform planning and decision making, an increasing amount of information should be provided to those, using 3D data have quite an impact on this process.

Robot Vision and 3D Grasping Robot vision refers to the capability of a robot to visually perceive the environment and interact with it [3]. Having perception of what is around is not trivial, the realization of a robot perception system is fundamental for having robots behaving in an intelligent way, interacting with

objects and people in realistic scenarios. An example of a robot complex interaction is object manipulation: a robot, once chosen a target object, must control its gripper moving it (in terms of 3D position and orientation) according to the position and structure of the target object. Several factors affect grasping process: object geometry, toughness or mass, gripper geometry, materials friction, possible interactions with other objects in the neighbourhood, *etc.* While object recognition and categorisation works reasonably well in the 2D domain, 3D vision is mandatory for building robot perception systems, enabling those to 'consciously' interact with real-world objects, performing everyday manipulation tasks in realistic human environments. Mousavian *et al.* in [4] tackled the problem of unknown objects grasping with a learning-based 3D vision framework whose goal is to automatically predict stable grasping regions of 3D point clouds.

Autonomous Vehicles Self-driving cars are equipped with several sensors (including LiDAR 3D scanners, radars, cameras, radars, and ultrasound imaging) in order to perceive what happens around them at different semantic levels and capacities. However, information is not knowledge, the difference between simply gathering information on what's around and getting some understanding of it is huge. Intelligent algorithms needs to be developed in order to exploit the gathered information for taking actions and react to stimulus from the world outside, this is exactly where AI comes into play. Vision systems in autonomous vehicles has to solve several tasks (including road detection, on-road object detection *etc.*) to successfully mature a perception of the surrounding environment. Additional information provided by 3D data is essential and is greedily exploited in building this environment perception. 3D data from LiDAR scanners provide multiple advantages with respect to 2D images, overcoming one of the main drawback in case of 2D-representation, *i.e.* performance degradation under different illumination and weather conditions.

1.3 Motivation

The novelty of this work consists in the introduction of *cross-domain adaptive learning* to 3D vision. A large amount of recent literature focusing on images has shown how the lack of data annotation and the possible domain shift issues can be alleviated by being able to integrate different source of knowledge in the learning process. Domain Transfer exploits pre-trained models as initialization when the number of training data is scarce. Domain Adaptation leverage transductively on the unlabeled target data to align training and target distributions. Domain Generalization elaborates optimization methods and regularization tools that lead to a better learning equilibrium conditions on the training data and reflects on

more reliable results on samples from new and variable testing conditions. Self-supervised tasks have also shown to be a helpful support. They are defined on unsupervised data, by neglecting part of the original inherent sample information (orientation, relative part position, color) and then using it as annotation for supervised learning. Tasks like rotation recognition or jisgaw puzzle solution can be easily used as pre-text warm up procedures for transfer learning in 2D.
When moving to 3D data, however, the design of self-supervision tasks is more difficult and less explored. Moreover, the need to deploy this models in real-world settings moves the problem from the controlled transfer learning setting on a single test problem to real generalization for any novel domain without the need of costly relearning procedures. Despite the remarkable results achieved for object recognition by operating on standard RGB images, there are inherent limitations due to the loss of data caused by projecting the 3-dimensional world into a 2-dimensional image plane. Further on, while 2D images have a unified representation as pixels array, 3D data can be modelled with multiple representations such as point cloud, mesh, volumetric field, multi-view images and parametric models, each fitting a specific application scenario. The existence of multiple acknowledged representations translates into a non-trivial challenge for 3D understanding, not only, since each representation has specific properties it is impossible to develop a unified deep learning architecture managing all them. The advent of PointNet [5], which processes irregular point cloud data by neural networks directly, draws attention to the 3D point cloud representation, making it the most widely used for 3D shape learning. Such network provides end-to-end classification and segmentation without the memory overheads of voxel grids or the potential loss of information from 2D image representations. Not only, with [6] a new research field named *Geometric Deep Learning* is opened, investigating the learning from non-Euclidean structured data as point clouds, meshes, manifolds, graphs. This context reveals the need for efficient 3D deep learning approaches, in addition there is huge industry interests: robotics, autonomous driving, virtual/augmented reality, smart manufacturing *etc.* All this highlights the importance of 3D vision for having intelligent systems reasoning about 3D space more effectively.

1.4 Contribution

We see this work as a first attempt to push the limits of 3D deep learning towards more realistic scenarios, with less stringent constraints in complexity of learning models and annotation resources towards a speed up of real world understanding for autonomous intelligent systems. 3D data became widely available and always more spread, not only, thanks to the rise in computational power researchers start looking at their potential. Although several representations of 3D data exists point

clouds represent one of the most valuable, standing out for availability, compactness and robustness. At the same time, learning from point clouds is definitely not easy due to the lack of an ordered regular structure. The recent success of PointNet [5] demonstrated that is possible to consume point clouds extracting feature representation in an efficient way, encouraging researchers to move in this direction. In this context 3D domain represents a golden opportunity for AI researchers, overall it is still terra incognita. Recent progress achieved by 2D deep learning in Domain Adaptation and Domain Generalization enabled deep learning models with better generalization abilities, this also facilitated the deployment of these models to real world settings. Unfortunately, those achievements are not easily transferable to 3D domain.
Our work aims to define a methodological approach for Domain Adaptation and Domain Generalization in the 3D domain. We meet the real world challenges for 3D shape understanding proposing our recipe, highlighting the importance of methods able to deal jointly with labeled and unlabeled data possibly coming from different domains. In this direction we proposed a multi-task approach combining supervised and self-supervised learning with compelling results in all the considered scenarios over several real world and synthetic datasets.

Chapter 2

Related Works

How to use powerful deep learning tools for multiple task accomplishment (first and foremost object classification and part segmentation) on point clouds is an extremely active area of research. One of the first deep learning models working with 3D point clouds was proposed by Qi *et al.* [5] and is called PointNet. It combines symmetry and spatial transform functions to learn point-wise features which are then aggregated into global representations. The follow-up work, namely PointNet++ [7], further introduce a hierarchical combination of different PointNet modules. More recently, PointCNN [8] introduced a transformation mapping shape vertices to a canonical space where their order is preserved and therefore allowing for the application of traditional convolutional operators on them. Many other solutions have also been published with the aim of making it possible the use convolutional filters on point clouds either in the spatial [9, 10, 11] or in the spectral domain [12, 13]. *Self-supervised learning* is a framework recently achieving a large attention in the 2D computer vision community. It deals with originally unlabeled data for which a supervised signal is obtained by first hiding part of the available information and then trying to recover it. This procedure is generally indicated as *pretext* task and possible examples are image completion [14], colorization [15, 16], relative position of patches [17, 18], rotation recognition [19], identification of synthetic artifacts [20] or image clusters [21]. Several applications also consider multi-modal data, involving depth [22], sound [23], motion-based segmentation, video temporal ordering [24] and physical interactions [25]. The solution of the pretext task capture high-level semantic knowledge from the data so that the learned representation can be transferred to other *downstream* tasks as a powerful warm-up initialization. The notion of self-supervision appears extremely relevant also for 3D structures, both to capture internal local information at a good neighborhood resolution before combining it with global shape knowledge, and to get information on context and surfaces. Recently, several works dealing with 3d

representations have proposed autoencoder-based architectures that search for self-organizing maps [26], or extract latent point capsules [27]. Other approaches either learn to deform a 2D grid onto the underlying 3D object surface [28], or directly learn to generate the surface of 3D shapes. The recent work of [29] split a point cloud into a front and a back half from several angles and train a model to predict one from the other. A network to verify whether two randomly sampled parts from the dataset belong or not to the same object is presented in [30], while [31] proposes to reconstruct point clouds whose parts have been randomly rearranged. In [32] a recurrent neural network (RNN) is trained to predict the next point on a sequence created by a space filling curve. Finally, reconstruction, clustering, and self-supervised classification are combined together in [33], defining a fully unsupervised multi-task approach for feature learning. The pretext and downstream stages define a particular case of *Transfer Learning* (TL) where unsupervised data is exploited to support supervised learning on a collection and task of interest. In many real-world applications, despite unlabeled data may be freely available, their distribution can significantly differ from that of the supervised data at hand rising the extra issue of how to deal with a *domain shift* for which the transfer procedure may backfire. A similar problem holds also when the unsupervised collection is not an extra source of knowledge, but corresponds instead to the test on which we need to evaluate a supervised model. *Domain Adaptation* (DA) literature focuses on this scenario supposing that the unsupervised test data is available at training time. Many adaptive solutions have been proposed in the last years for 2D vision problems some of which involve feature alignment strategies with dedicated losses [34, 35, 36, 37], ad-hoc network layers [38, 39] and adversarial learning approaches [40, 41]. Several recent works also involve generative style transfer methods [42, 43], reconstruction penalties [44, 45, 46] or conditions on metric [47] and feature norms [48]. An even wider and more challenging problem is the one tackled by *Domain Generalization* (DG). In this case, the specific adaptation target is not provided at training time and the goal is learning a model robust to any kind of new domain shift that can appear during deployment. Only few works have shown good results in this setting, mainly considering feature alignment among multiple data sources when available [44, 45, 49, 50], data augmentation [51, 52] and meta-learning [53, 54]. Most recently, self-supervised learning has also shown promising results in the DA and DG scenarios [55, 56, 57]. Although the main focus of the aforementioned literature is object classification, the problem of dealing with consistent domain shifts extends also to object detection and semantic segmentation where several works have investigated the gap between synthetic and real data which is crucial for automotive and robotics applications [58, 59, 60]. Still, all the mentioned literature focuses on 2D images. Only one very recent work has started investigating DA for 3D point clouds, [61] just touching the problem for

moderate domain shifts and leaving open questions on how to deal with real world applications in large need of adaptive solutions as evidenced by [62].

Chapter 3

Background

3.1 Self-Supervised Learning

Most modern machine learning algorithms have the same drawback: the need for huge amount of labeled training data, which are significantly more expensive and time-consuming to obtain than raw unlabeled data. This drawback is emphasized when dealing with learning tasks such as semantic or part segmentation which require dense annotations. Self-Supervised Learning (SSL) is a particular form of unsupervised learning in which the data provide the supervision by itself. SSL have shown great potential, making the most of unlabeled data which are often largely available and 'free to use'. SSL does not depend on human annotations, instead exploits pseudo labels directly generated from the data [63]. In SSL a supervised signal is obtained by first hiding part of the available information and then trying to recover it. This procedure is generally indicated as *pretext* task, possible examples are image completion [14], colorization [15, 16], channel prediction [64], relative position of patches [17, 18], rotation recognition [19], identification of synthetic artifacts [20] or image clusters [21]. Several applications also consider multi-modal data, involving depth [22], sound [23], motion-based segmentation, video temporal ordering [24] and physical interactions [25].
Recent publications have also shown *extensions to 3D point clouds* for object and scene understanding as well as to 3D medical structures. Specifically [30] trains a network to verify whether two randomly sampled parts from the dataset belong or not to the same object and [31] proposes to reconstruct point clouds whose parts have been randomly rearranged. Clustering and autoencoder-based reconstruction are combined in [33], while in [32] an RNN is trained to predict the next point on a sequence created by a space filling curve. In [65] translational and rotational invariant volumetric features are learned by solving a Rubik's Cube. In all the previous cases, the pretext tasks are defined so that a high-level semantic

understanding is essential for their solution: the knowledge encoded in the learned deep feature representation is then transferred to other *downstream* tasks that are trained either with linear SVM on those features or by fine-tuning from the self-supervised models. Besides this sequential two-step procedure, recent works focusing on image recognition have proposed alternative approaches based on a multi-task framework [55, 56, 57] where supervised and self-supervised learning are combined together with the the latter supporting robustness of the former in case of limited annotated data and domain shift.

3.2 Multi-Task Learning

Multi-Task Learning (MTL) is an inductive transfer mechanism aiming to improve the generalization of machine learning models by learning multiple tasks in parallel. When dealing with multiple related tasks, training them in isolation may not be the best strategy. Instead, learning them in parallel can lead to better generalizing models because the MTL process takes advantage of the common information shared across the tasks to learn better features, availing of domain-specific information which are contained in related tasks training signals. Domain specific information of a task can act as inductive bias for others. In the MTL setting, the network backbone share a certain number of hidden layers between all tasks, in this way features learned by a task can help other tasks to better learn. A pretty common phenomena in human learning process, a person who has learnt riding a tricycle will do half of the work when learning to ride a bike because will exploit useful knowledge from the 'tricycle domain'. An example of a MTL architecture is presented in [66], where a standard neural network architecture with a single fully connected hidden layer is trained to solve 4 tasks in parallel 3.1. When multiple tasks share the hidden layers of the same neural network architecture, and only the heads of the network are task-specific, we refer to *hard parameter sharing*. The introduction of multiple parallel tasks has a regularization effect and reduce the overfitting, because the parameters of the shared hidden layers should have a representation capturing all of the tasks instead of specializing to a single one. Recent works [67, 68, 69] proposed a different way of doing multitask learning. Multiple tasks have their private model and their own parameters. At the same time different tasks parameters are interconnected between them, allowing one task with its own model to draw from other tasks parameters [70], we refer to this approach as *soft parameter sharing*. Although each task has its own model with its own parameters, the distance between the parameters of the model is then regularized in order to encourage the parameters to be similar.

Multi-Task in Geometric Deep Learning To the best of our knowledge, there are only few works in Geometric Deep Learning adopting a real end-to-end multi-task approach. In a very recent work, Hassani et al. [33] proposed a new architecture for unsupervised multi-task feature learning on point clouds. The architecture is composed by a first multi-scale graph-based encoder extracting point and shape features from the input point cloud. Three more decoders, each specialized in solving a specific task among Reconstruction, Clustering and Prediction, work on the feature extracted by the multi-scale decoder, jointly providing the architecture with a multi-task loss.

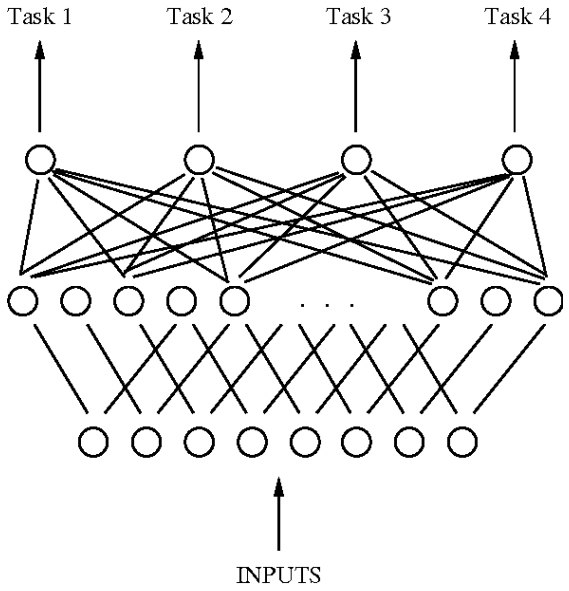

Figure 3.1. Schematic of Hard Parameter sharing MTL network with a single shared hidden layer. Figure source: [66]

3.3 Domain Shift

A distribution shift happens when a deep learning model, once trained on data from a specific distribution (source) is used to inference on data from a different but someway related distribution (target), in general target distribution shares the labels space with source. The consequence of domain shift is a model well performing on source domain but performing poorly on target domain, an issue well known in Computer Vision community. Dealing with visual distribution shifts is fundamental since living in a constantly evolving world. In real-world setting the distribution of test data (target) often differs from that of training data (source).

The endless number of real world domain shifts makes impossible to consider all of them in training source domain, thus making domain shift the main issue preventing in-lab trained deep learning models from being successfully deployed in real-world settings. Existing approaches for domain shift mitigation can be categorized into two main branches which are *Domain Adaptation* (DA) and *Domain Generalization* (DG), while in the 2d domain this problem has been largely discussed there are only few works doing the same in the 3d domain.

3.3.1 Transfer Learning

We define a domain $\mathcal{D}$ as the combination of a d-dimensional features space $\mathcal{X} \in \mathcal{R}^{\mathcal{D}}$, a marginal probability distribution $P(X)$ and a task $\mathcal{T}$. The latter ($\mathcal{T}$) in its turn is defined by a label space $\mathcal{Y}$ and the conditional probability distribution $P(Y|X)$.
Considering a single domain $\mathcal{D} = \{\mathcal{X}, P(X)\}$ with $\mathcal{T} = \{\mathcal{Y}, P(Y|X)\}$, supervised learning focus in learning the conditional probability $P(Y|X)$ from the samples $x_1,\ x_2, \ldots,\ x_n \in \mathcal{X}$ and the corresponding labels $y_1,\ y_2, \ldots,\ y_n \in \mathcal{Y}$.
When considering multiple domains, e.g. a source $\mathcal{D}_s$ and target $\mathcal{D}_t$, when these have different distributions, the machine learning model trained on source is not guaranteed to perform well on target one, especially if the two are significantly different. When source and target distributions are different but somewhat related it is possible to catch the 'related info' from the source domain $\mathcal{D}_s$ exploiting it to learn the target conditional probability $P(X_t|Y_t)$, this process is called Transfer Learning.

Definition 3.3.1. (Transfer Learning) *Given a source domain $\mathcal{D}_s$ and learning task $\mathcal{T}_s$, a target domain $\mathcal{D}_t$ and learning task $\mathcal{T}_t$, transfer learning aims to help improve the learning of the target predictive function $f_t(.)$ in $\mathcal{D}_t$ using the knowledge in $\mathcal{D}_s$ and $\mathcal{T}_s$, where $\mathcal{D}_s \neq \mathcal{D}_t$ or $\mathcal{T}_s \neq \mathcal{T}_t$* [71].

Transfer Learning can be divided into two main categories: homogeneous and heterogeneous. In *Homegenous* Transfer Learning setting source and target domains, although having different marginal distribution due to domain shift, are represented in the same feature space: $P(X_s) \neq P(X_t)$ with $\mathcal{X}_s = \mathcal{X}_t$. Homogeneous TL focuses in reducing the gap between the data distributions of the two domains in cross-domain transfer. Differently, in TL *heterogeneous* setting, source and target data can have different representations $\mathcal{X}_s \neq \mathcal{X}_t$. [71] further categorizes TL approaches into three main groups: inductive TL, transductive TL and unsupervised TL. In *inductive TL* source task and target task are different but related ($\mathcal{T}_s \neq \mathcal{T}_t$), source and target domain can also be the same.
In *transductive TL*, source and target task are the same $\mathcal{T}_s = \mathcal{T}_t$ while source and target domains are different: $P(X_s) \neq P(X_t)$ with $\mathcal{X}_s = \mathcal{X}_t$ or $\mathcal{X}_s \neq \mathcal{X}_t$. In general

source domain is labeled while no labeled data are available in target domain. Last setting is *unsupervised TL*, as for the inductive case, the source task is different but related to the target task. However, in this setting, no labeled data are available, nor in source nor in target domain.

3.3.2 Domain Adaptation

Definition 3.3.2. *Domain Adaptation* (DA) consists in learning a discriminative classifier or other predictor in the presence of a shift between train and test distribution [72].

DA can be seen as a slightly relaxed transductive TL setting in which task and label space across the different domains are common, $\mathcal{T}_s = \mathcal{T}_t$ and $\mathcal{Y}_s = \mathcal{Y}_t$. Digging into DA setting: we consider source domain(s) and an unseen domain of interest (target). Source and target domains are different, $P(X_s) \neq P(X_t)$, but somehow related. In general a huge quantity of supervised data is available from source domain/s. In addition to supervised data, when dealing with DA setting, are also available some unsupervised training data in the domain of interest (target). As expected, because of the differences between source and target domain distributions, a model trained only on the source supervised data will perform poorly on target domain. Domain Adaptation aims to solve this problem taking advantage of target domain unlabeled data.

DA through self-supervision DA can be performed by learning auxiliary self-supervised task(s) on both domains, source and target, simultaneously. The main idea behind this is that solving self-supervised tasks on the destination domain samples induces alignment between the source and target distributions. Self supervised tasks are expected to bring the source and target domains closer together along the direction relevant to the task. Jointly train the main task classifier on supervised source domain data along with self-supervised tasks on target unlabeled data is shown to produce better models with increased generalization abilities on target domain. Possible examples of self-supervised tasks are image completion [14], colorization [15, 16], relative position of patches [17, 18], rotation recognition [19], identification of synthetic artifacts [20] or image clusters [21].

DANN Ganin *et al.* in *Domain-Adversarial Neural Networks* [72] propose an approach to Domain Adaptation for deep learning architectures, the resulting framework is presented in Figure 3.2. In DA setting a large amount of labeled data from the source domain is available, together with large amount of target domain unlabeled data. The DA main goal is producing deep features that are: (1) discriminative for the main learning task on the source domain and (2) invariant

with respect to the shift between the domains. DANN strategy achieve this by training a deep neural network architecture with two discriminative classifier in a multi-task way: (1) label classifier, solving the primary classification task, used at both train and test time; (2) domain classifier, working only at training time, has to predict whether a sample is drawn from source or target distribution. The secondary domain classification task has to predict the belonging domain for each sample, thus represents a self-supervised binary task (source and target being the labels), perfectly consistent with DA setting. During training the underlying deep learning model parameters are optimized in order to minimize the label prediction loss (1) and to *maximize* the domain classifier loss (2). This is achieved by connecting the only domain classifier block to the feature extractor via a gradient reversal layer (GRL) that multiplies the gradient by a certain negative constant during the back-propagation based training. Maximizing the domain classifier loss during training encourages domain-invariant features to emerge in the course of the optimization. Extracted features will be as indistinguishable as possible for the domain classifier.

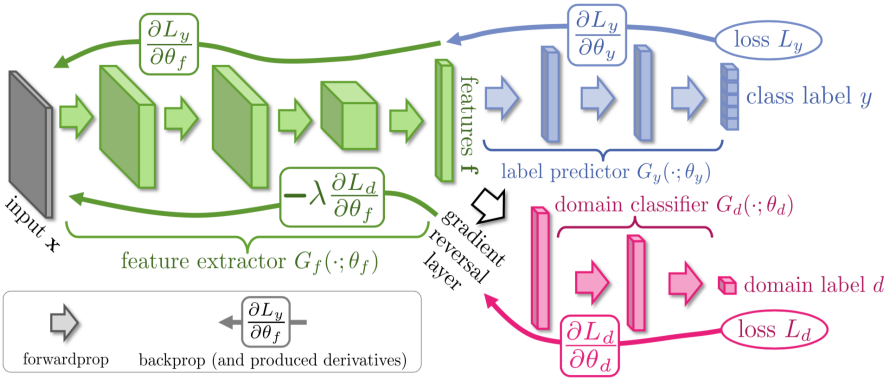

Figure 3.2. DANN architecture consists of three parts, a deep feature extractor (green), a deep label predictor (blue) and a domain classifier (red). Domain classifier is connected to the feature extractor via a GRL module, the latter ensuring that the feature distributions over the source and target domains are made similar. Figure source: [72]

3.3.3 Domain Generalization

A model trained with source domains data may perform poorly on a new and unseen target domain with a different distribution from the source one. This represents a huge limitation for deep learning models in practical applications. In real-world are common situations in which a model, once deployed, has to face

with a distribution different from the training one (e.g. different background, light condition etc.). Developing algorithms that are invariant to dataset bias is a problem widely acknowledged in machine learning community. Domain Generalization setting assume to learn from multiple source domains and to extract a *domain-agnostic* model. With domain-agnostic we intend a model performing uniformly well on unseen target domains with different distributions [73]. We emphasise that DG (in contrast to DA) addresses this problem by leveraging only on the source domains labeled data, aiming to learn a universal representation when no target data are available [74]. Most of works in literature addressed the DG problem in a data-driven way, exploiting labeled data from multiple source domains, aiming to learn from these a universal representation generalizing well also to unseen target domains. The basic idea is that minimizing the distribution variance among multiple source domains should also provide a domain agnostic representation. Data-driven approaches drawback is the risk of overfitting source domains, the learned representation may generalize well to any of the source domains but poorly to an unseen target domain. Starting from the assumption that each domain is biased, composed of an intrinsic globally shared factor and a domain specific component, Khosla *et al.* in [75] explicitly takes into account for bias when combining multiple source domains. Their method consists in learning two weight sets: (1) bias vectors associated with each individual dataset and (2) *visual world weights* that are common to all datasets. Visual world weights are obtained by factoring out from each dataset its associated bias, these are the one with good generalization ability. [55] propose a novel approach to DG (and DA) called Jigsaw Generalization. In this work supervised and unsupervised inherent signals from the images are jointly exploited, showing that generalization across visual domains can be achieved in a multitask way, by learning at the same time to classify and image parts spatial relations. Demonstrating that learning specific type of invariance leads to models with increased generalization ability.

3.4 A leap into 3D and Real World

Nowadays machine learning methods have achieved remarkable success, especially under the assumption that source and target data are sampled from the same distribution. Unfortunately in real-world applications, this assumption is often violated and, as a result, the trained system will perform poorly on target domain data. This issue can be mitigated with Domain Adaptation (DA) or Domain Generalization (DG). DA tackles this issue by aligning source and target distributions exploiting unlabeled target data samples. However, DA is contingent upon the availability of unlabeled samples from the target domain for guiding the adaptation procedure, unfortunately this is not always guaranteed, especially in real-world

applications. A more challenging setting is DG, this setting does not depend on the availability of target samples, works by training the model on multiple source domains data, guiding the learning to capture domain agnostic information. The resulting models should be discriminative for the trained task and, at the same time, insensitive to domain shifts.For deploying deep learning models in real world settings these generalization abilities are crucial, if a robot has to be deployed to people's homes has to be robust to the inherent distribution shift between the training setting and any random house.While this problem has been largely discussed in the 2d domain, the same is not true for the 3d non-Euclidean structured domain. To the best of our knowledge only few works deal with the Domain Adaptation or the more challenging Domain Generalization setting directly on *point clouds*. The recent PointDAN [61] represents a first attempt to Domain Adaptation with point clouds, proposing a framework which jointly aligns the global and local features. Moreover, most of the previous works only deal with the *train on synthetic* $\rightarrow$ *test on syntetic* scenario which is not representative of real world challenges. ScanObject [62] recently raised this issue, proposing a new *real world* 3d point cloud benchmark dataset, demonstrating that Object Classification is still a challenging task if performed on this type of data. This context reveals the need for efficient 3d deep learning approaches, with this work we propose a *hard parameter sharing multi-task* architecture highly exploiting self-supervision. In both DA and DG settings, we analyzed the problems arising when shifting from synthetic to real world distribution (or vice versa) and we proposed a robust approach for dealing with real world challenges. It is interesting to note that few multi-task architecture have been proposed in geometric deep learning literature, infact most of previous works use self-supervision on huge datasets generating an initial set of weights and then train (in a fully supervised manner) the task of interest using those weights as initialization [32, 31]. Robotics, autonomous driving, virtual/augmented reality, smart manufacturing *etc.* are eagerly waiting for robust 3d point cloud learning methods capable of dealing with real world challenges, our work is the first push in this direction.

Chapter 4

Geometric Deep Learning

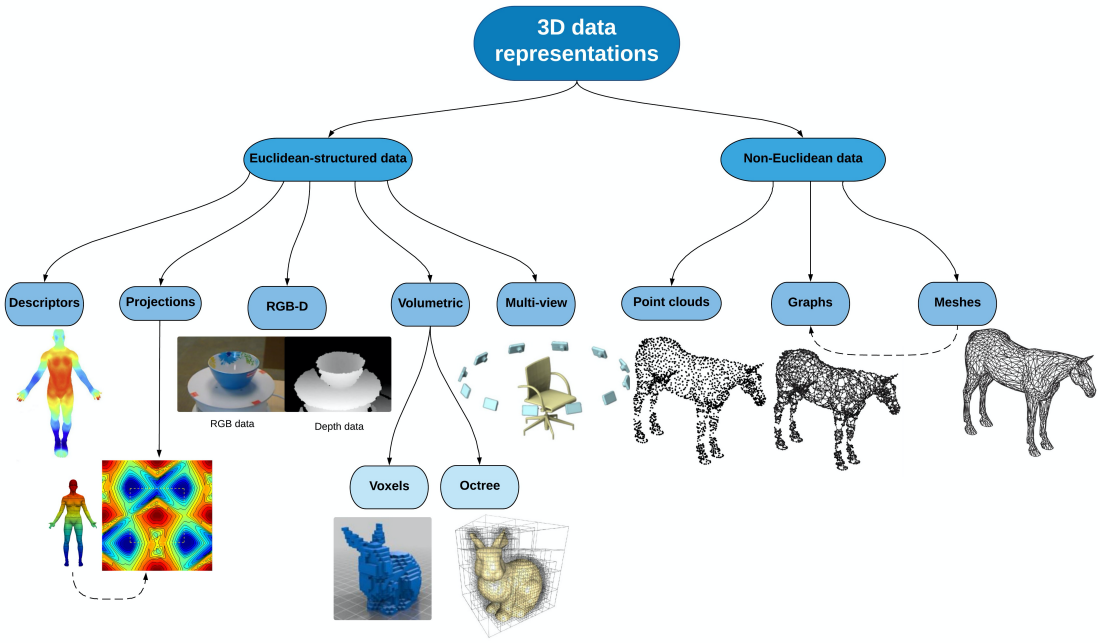

Figure 4.1. Summary of all 3d data representations. 3d representations are further split into Euclidean structured and non-Euclidean. Figure source: [76]

Deep Learning has boosted computer vision field, with the advent of deep Convolutional Neural Networks (CNN) impressive results have been achieved on several 2d domain tasks. The success of DL on images and the increasing availability of 3d data has motivated a *data-driven approach* to learn features also from this type of data , aiming to exploit the higher information quantity in 3d data to achieve the same promising results - if non better - also in this challenging

domain. However moving deep learning architectures to 3d is not trivial due to the complex geometric nature of 3d objects and the several different types of 3d representation existing. RGB images, audio, text, and many other types of data fed to CNN are all Euclidean structured data; shapes, manifolds, graphs, point clouds lack of an Euclidean structure. Although the CNN achievements and results in literature, traditional CNNs can't handle non-euclidean structured data. This limitation arises from the use of filters that are location invariant, those filters can be applied only on input data with regular structure.
Shape learning is defined as the learning of a mapping from input geometrical signals to features [77]. The specific representation of geometrical signals defines: (1) the learning architecture to be used, (2) the quantity or richness of information preserved by the representation. The latter being a bottleneck to the downstream learning process. For all the reasons discussed, while applying classic deep learning techniques to euclidean structured data (also in 3d, *e.g.* RGB-D) is immediate, elaborating a data-driven approach for learning features from non-Euclidean structured data is not straightforward and poses new challenges.

4.1 Euclidean Data

Euclidean data have an underlying grid structure allowing global parametrization and a common system of coordinates. In the case of 3d representation, the euclidean structure makes easy extending 2d deep learning methods to 3d, this is done by simply using the same convolutional operator as defined in the 2d domain. Examples of 3-dimensional euclidean data are RGB-D data, multi-view images, *etc.*

RGB-D data RGB-D data are combination of RGB image (I_{RGB}) and corresponding depth map (I_D), the latter contains information on the distance from the scene to the camera. RGB-D images have become popular thanks to the availability of low-cost RGB-D sensors (*e.g.* Microsoft Kinect), defining what is commonly referred as 2.5D. A huge number of RGB-D datasets exist, especially if comparing to datasets of other 3d representation such as point clouds. Besides being inexpensive and largely available, by using RGB-D data remarkable results has been achieved in several tasks including: scene reconstruction, pose regression, recognition. Since having an euclidean-structure, RGB-D data can be processed by using traditional 2d deep learning methods.

4.2 Non-Euclidean data

Shapes are represented in a non-Euclidean structured domain. Examples of other non-Euclidean data are: point clouds, manifolds, mesh and graphs. Their non Euclidean nature implies that fundamental data properties as: (1) global parameterization, (2) common system of coordinates, (3) vector space structure, (4) shift invariance, are not followed. This represented the major obstacle that so far has precluded the use of successful deep learning methods such as CNN or RNN on non-Euclidean geometric data. Convolutional operator which is the key point of CNN success in the Euclidean domain is not well defined in non-Euclidean, making the translation of CNN and deep learning to non-Euclidean domain not immediate. Concretely, generalizing DL models to non-Euclidean data involves finding a non-Euclidean counter part of the basic building blocks composing these models, *e.g.* replacing Euclidean-defined convolutional operator with pooling operator [5]. Bronstein et al. in [6] specific focused on non-Euclidean data referring to as *Geometric Data*, further on they also coined the term *Geometric Deep Learning* indicating situations in which Deep Learning is applied to the previous defined Geometric Data. In this work concrete methodologies to transfer Deep Learning techniques and achievements to non-Euclidean data domain are proposed, the work focus is on generalizing CNNs to non-Euclidean structured domains enabling deep learning on graphs, manifolds, meshes and point clouds.

4.3 Point Clouds

Point cloud (pc) is a type of non-Euclidean data providing a compact and expressive representation of 3d geometry. Point clouds are an irregular format by definition, point positions composing the 'cloud' are continuously distributed in the space, any permutation of their ordering does not change the spatial distribution. Learning from point clouds is not trivial due to their inherent irregularity. Previous approaches transformed such data to Euclidean-structured: regular 3D voxel grids or collections of images. This, however, renders data unnecessarily voluminous and causes issues: (1) introduction of representation errors, (2) inevitable reduction of the information content. We formally define point cloud Φ as an *unordered set* of vectors x_i:

$$\Phi = \bigcup_{i=1}^{i=N} x_i, \text{ with } x_i \in \mathbb{R}^D \tag{4.1}$$

A vector x_i represents a point in some space. In general, we are not constrained to 3-dimensional space, it is possible to add more features to each point (e.g. color,

normals, etc.), moving to a higher dimensionality space. Due to their unstructured nature point clouds represent an irregular non-Euclidean data representation. Deep learning models consuming pcs must be *permutation invariant*: given a Point Cloud, points permutation should not alter task (recognition, segmentation, retrieval *etc.*) result.

$$f(x_1,\ x_2, \ldots,\ x_n) \equiv f(x_{\pi 1},,\ x_{\pi 2}, \ldots,\ x_{\pi n}),\ \text{with}\ x_i \in \mathbb{R}^D \tag{4.2}$$

In our learning pipeline input point clouds, before being fed in input to the used network, are: (1) centered, (2) unit cube scaled and (3) random sampled to fixed num points. Although possible designing special convolution operators working in non-Euclidean domains consuming point clouds, rotation invariance is often not guaranteed, leading to neural networks generalizing poorly to arbitrary axis rotations. PointNet [5] tries to fulfill this constraint by using a Spatial Transformer Network (STN) aligning point clouds in input before feeding them to the feature extractor. To the best of my knowledge only few works propose point cloud networks that are really Rotation Invariant. Instead, most point cloud DL networks exploit approaches similar to PointNet (using STN) or use data augmentation techniques in order to be robust to different input pc orientations. The few recent proposing point cloud rotation invariant convolutional operators, as [78], deal with low-level geometric features such as distances and angles which are by definition rotation invariant.

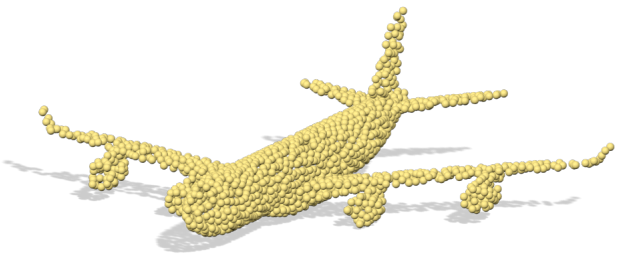

Figure 4.2. Point Cloud representing Airplane CAD from the ModelNet40 dataset. Points have been uniformly sampled to 2048 on the mesh faces according to their face area.

4.4 Geometric Deep learning Tasks

The impressive results achieved by computer vision in the 2d domain and the need for AI-based systems successfully deployed in real-world environments motivated

the push for robust vision algorithm also in the more challenging 3-dimensional domain. In this section we highlight the most important vision tasks and the relative state-of-the-art in the field of *geometric deep learning*. In this work we mainly used point cloud as ours 3d data representation, encouraged by its spread in the deep learning community and the several recent works dealing with. However, how to use powerful deep learning tools for different task accomplishment on point clouds is an extremely active area of research that originally started with PointNet [5].

Object Classification Object Classification is a fundamental computer vision task. Given a 3D object shape, the goal is identifying the class or category to which the shape belongs. Formally, a classification algorithm provides a function:

$$f : \mathbb{R}^n \to \{1, \ldots, k\} \tag{4.3}$$

assigning to each n-dimensional input vector x an integer number $y = f(x)$ corresponding to one of the possible k categories. When working with non-Euclidean data as point clouds the widespread classification approach is the one proposed by PointNet [5]. Classification is performed directly on unordered point sets rather than passing to an intermediate regular representation. PointNet operates on each point independently and then applies a symmetric function to accumulate features, the use of a symmetric function let PointNet achieve permutation invariance property. The main PointNet drawback is that largely treats points independently at local scale, neglecting geometric relationships among points. Several extension of PointNet has been released, [7] improves PointNet taking advantage of local structures formed by the metric space. Points are first partitioned into local regions that overlap by the distance metric of their space, then features are extracted with a hierarchical 'fine to coarse' process. Common testbed for object classification performances is the ModelNet40 dataset [1] on which geometric deep learning based methods have achieved the best performance. [11] propose a geometric deep learning backbone achieving state-of-the-art performance on this dataset with an overall accuracy of 93.5pp using 2048-points. However, shapes in ModelNet40 are synthetic CAD models of isolated object with no background, despite the near superhuman performance achieved on synthetic data we argue that shape classification is still a challenging task when dealing with real-world object scans.

Part Segmentation Part Segmentation task consists into dividing the input shape into meaningful parts. We formalise the Part Segmentation task on 3d point cloud input shapes as the problem of assigning each vertex p to the part of such shape to which it belongs to. Typically, part segmentation is modelled as a point-wise classification problem where each shape part is represented by an integer

label in $\{1, \ldots, P\}$. The common benchmark dataset for Part Segmentation on Geometric Data in ShapeNet Part [79], it provides per-point labels for each shape, indicating which points belong to the region of interest. Fine-grained tasks like part segmentation require detailed structural information, following [5], the quality of the predicted part segmentation is evaluated in terms of the mean Intersection-over-Union (mIoU) metric. The mIoU of a shape with P parts is defined as the average over P parts of the IoU between ground-truth and predictions of each part. The mIoU of a category is defined as the average of the mIoUs of the shapes it contains. In Figure 4.3 we show the Part Segmentation result on a lamp category shape from the ShapeNet Part dataset. A Geometric Deep Learning model predict for each point of the shape a part prediction, representation has been obtained by assigning a different color to each predicted part.

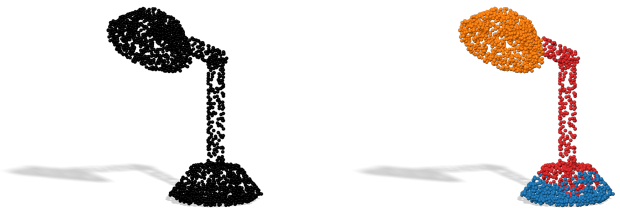

Figure 4.3. Part Segmentation on ShapeNet Part, example for lamp category object. Right side figure represents ours GDL model (trained with only 1% of supervision) parts prediction.

Semantic Scene Segmentation Semantic Scene Segmentation is the task of splitting an input scene into semantic object classes, rather than object part labels. Scene understanding is a core computer vision task, in both 2d and 3d domain. This problem is highlighted by the fact that an increasing number of intelligent systems have the urge for context understanding. Semantic segmentation achieves fine-grained inference by making dense predictions inferring for each point of the cloud its enclosing object region.

4.5 PointNet

PointNet [5] is the first end-to-end neural network architecture directly consuming point clouds, enabling the solution of multiple task on unordered set of points: Shape Classification, Semantic Segmentation and Part Segmentation. The authors

provide a network capable of working directly with (x, y, z) coordinates and, if needed, with additional dimensions; network architecture is shown in Figure 4.4. PointNet achieve Permutation Invariance property by using max pooling symmetric function. N.B. Pooling layers represent the equivalent of 2d convolutional operators in the pc non-Euclidean domain. Considering an input Point Cloud with n (x,y,z)-points, max pooling function is used after the n input points are mapped to higher-dimensional space. The function output represents a global feature vector aggregating the captured information from the n input points, better said, a *global descriptor* for the entire shape. Another property to consider when dealing with Point Clouds is Transformation Invariance. Shape classification (but also segmentation and other tasks) shouldn't depend on the (x,y,z)-rotation degree applied to the input shapes. PointNet aims to be robust to this by introducing two data-dependent spatial transformer networks (STN3d and STNkd). The first one (STN3d) attempts to align different input shapes before being processed by the PointNet itself, for each shape in input an affine transformation matrix is predicted and applied to shape input points coordinates using dot product. The second alignment network (STNkd) produce an alignment in the feature space, predicting a feature transformation matrix to align different input point clouds features. Classification and segmentation branch share most of the network architecture (main features extractor). The Classification branch takes in input global features and outputs class scores. Differently, the segmentation branch takes in input the concatenation of local point features (network output after STNkd feature transformer) and global features (network output of max pooling), producing in output per point scores.

Part Segmentation Arch. The PN Part Segmentation network is a slightly modified version of the main basic architecture. Object shapes from different categories have different parts, *e.g.* a 'chair' won't have the same parts of an 'airplane'. In order to overcome this and train on shapes from multiple object categories a one-hot vector, indicating the input shape category, is introduced. For each shape, the category one-hot will be concatenated with the shape features from the max-pooling output. To boost performance, the capacity of some layers is increased and skip-connections have been introduced to collect and concatenate local point features from different layers. For each shape, a features vector is obtained by concatenating: (1) its category one-hot, (2) the max-pooling output and (3) local point features from different backbone layers; the obtained vector is then fed to the segmentation network which outputs part scores.

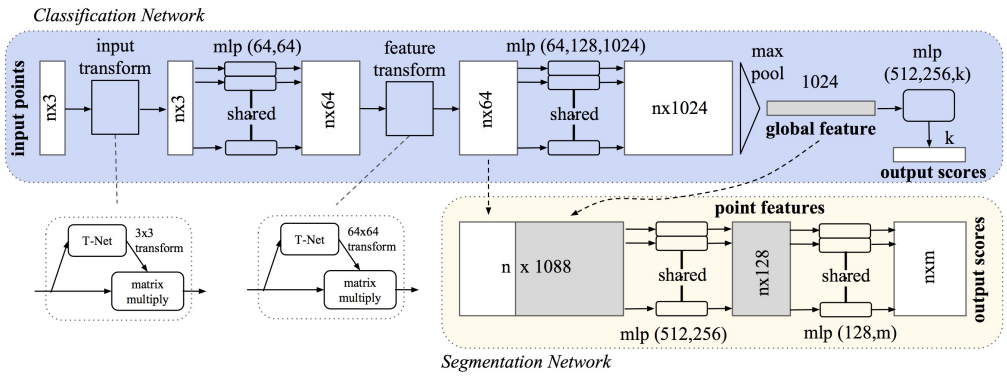

Figure 4.4. PointNet architecture: classification and segmentation networks. Input is raw x, y, z point cloud data of size $n \times 3$. raw Point Cloud. 3×3 transformation matrix is applied to input Point Cloud. If enabled also a 64×64 transformation matrix is applied regularizing features from different point clouds. Point features are aggregated by max pooling. All layers (except last one) include batch normalization and ReLU; Dropout is used in the final MLP. Image Source: [5]

4.6 PointNet++

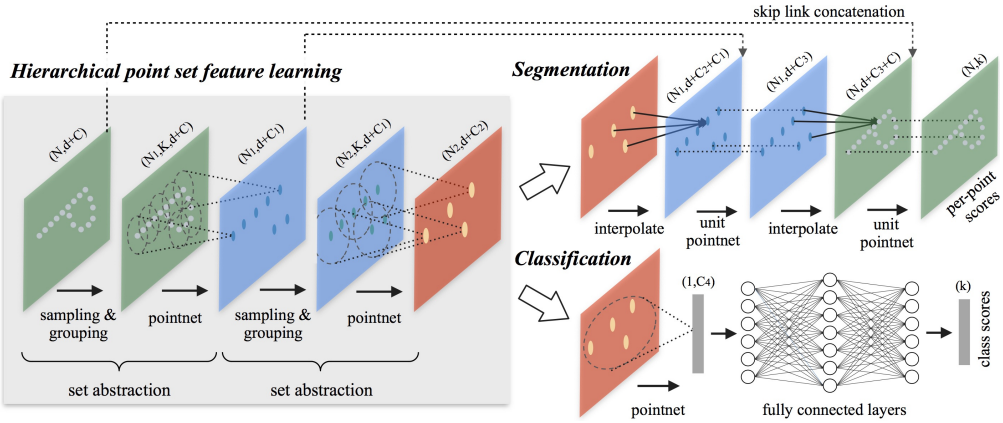

Figure 4.5. PointNet++ architecture. Hierarchical feature learning is introduced to learn features at various scales. Sampling and grouping layers define and sample neighbourhoods at various sizes which are fed into a PointNet module architecture for local pattern extraction. Image source: [7].

PointNet++ [7] is a hierarchical feature learning framework on point clouds; architecture is shown in Figure 4.5. This architecture works by applying PointNet

recursively on nested partitions of the input point set. It also proposes novel layers with the aim of learning from point clouds with non-uniform densities. Two key actions are iterated during this process: partitioning and abstracting sets of points or local features by means of a local feature learner. Given an unordered set of points, first step is partitioning it into overlapping local regions. Local features, capturing fine geometric structures of a local neighborhood, are extracted from each partition. These features are further grouped into larger units and then processed in order to produce higher level features. This process is repeated until we obtain the features from the whole cloud. Digging into details, a partition is represented by a 'neighborhood ball' in the underlying Euclidean space. Each neighborhood ball is completely defined by only two parameters: centroid location and scale. Since it is mandatory to cover the whole set of points, some points of the point cloud are promoted to centroids, these points are selected by using a farthest point sampling algorithm. The hierarchical way in which PointNet++ works reminds us of CNNs. However PointNet++ local receptive fields depend on both the input data and the metric, because of this should be more effective. PointNet++ exploits PointNet as a local feature learner. PointNet, in this new architecture, takes the role of a functional block, capturing higher level representations from local point sets or features for local pattern learning.

Chapter 5

Datasets

In this chapter we briefly introduce the main datasets used in this work. We mostly used datasets which are common benchmark for 3D-Object Classification and Segmentation.Recent 3D object classification methods reported state-of-the-art performance on CAD model datasets such as ModelNet40.However, CADs are high quality synthetic models, not alway near to what real-world scans are, we argue that 3D-object classification and segmentation are definitely not trivial when performed in real-world settings.

5.1 Synthetic vs Real-World data

The recent availability of large-scale synthetic 3d datasets motivated the recent advances in 3d deep learning, especially in Object Classification and Segmentation tasks, most of the proposed learning methods [5, 7, 11, 8] have been evaluated using purely synthetic data, focusing on isolated object models, a complete evaluation of those methods on real world scans is essentially missing.

The 3d technology evolution lead to cheaper and more reliable sensors, a growing up number of different types of 3d sensors are now easily deployable on intelligent systems such as robots or autonomous vehicles. In this context the amount of 3d real-world data spread, drawing the DL community attention to robust deep learning methods performing well also on this type of data. Real-world data are significantly different from CADs due to the presence of background, occlusions, reconstruction errors. In real word scans objects are often in under-segmentation situation, meaning that in most of 3d scans not only is present the target object but also background and, maybe, pieces from interacting or nearby objects. An example is represented by real-world scan of a 'pen' lying on a desk. The presence of background and pieces from nearby objects is a double-edged sword. If on the one hand it is possible take advantage of this context information, on the other

hand this information if not properly exploited can easily distract the learning architecture leading to worse results. Due to the inherent differences between real and synthetic 3d data, such as noise and occlusion patterns, deep learning models trained on synthetic perform bad when test is performed on real data, this highlights the need for robust deep learning models generalising well also to real data.

5.2 Modelnet

ModelNet (Wu et al. [1] , 2014) is a a large-scale 3D CAD models dataset for 3d object recognition and classification. Represents one of the most well-known and commonly used 3D dataset. The authors, from Princeton University, acquired a huge amount of 3D CAD models, manually checked and categorized each one. They removed irrelevant objects from each model (e.g, background, floor, thumbnail image, person standing next to the object, etc) and discarded unrealistic or duplicate models. This work led to a 151,128 3D CAD models dataset, each CAD belonging to only one of the 660 overall object categories. ModelNet provides the 3D geometry of the shape in form of point cloud, without any information about the texture. The multitude of works in which ModelNet dataset has been used for recognition and retrieval tasks highlights the spread of DL methods for learning such 3d unstructured data. From ModelNet two subsets are available and commonly used: ModelNet40 and ModelNet10, with 40 and 10 object classes/categories respectively. Note that the ten classes of ModelNet10 are a just a subset of the forty from ModelNet40. ModelNet40 contains 12,311 CAD models divided into 40 total categories; synset of categories in Table 5.1. The authors provides an official training and testing split. The number of training samples is 9, 843 while there are 2, 468 samples for testing. CAD Models have not the same orientation and have widely different scales. Before performing our experiments, when using ModelNet40, we will firstly scale each shape to Unit Cube. ModelNet10, being a ModelNet40 subset, consists of 4899 CAD models from 10 classes. Unlike for ModelNet40, ModelNet10 CAD models are orientation aligned along their gravity axis. However, since models are not aligned w.r.t scale we will perform also for this subset unit cube scaling. ModelNet10 training split consists of 3991 CAD models while 908 CAD models are reserved for testing.

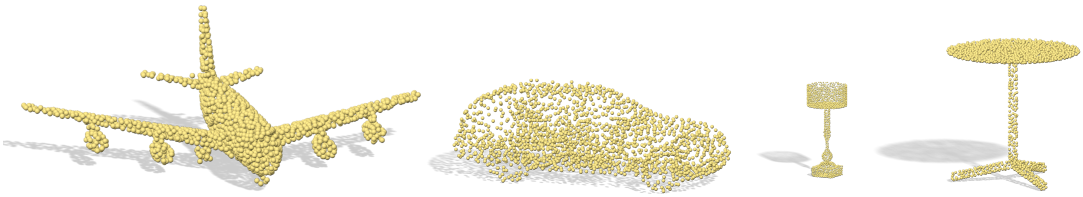

Figure 5.1. Rendering of shapes from ModelNet40 dataset, from left to right: air, car, lamp, table.

Table 5.1. ModelNet40 categories distribution, CAD models are not equally distributed between categories. Categories in ModelNet10 subset are bold.

Category	**Train**	**Test**	**Category**	**Train**	**Test**
airplane	626	100	laptop	149	20
bathtub	106	50	mantel	284	100
bed	515	100	**monitor**	465	100
bench	173	20	**night stand**	200	86
bookshelf	572	100	person	88	20
bottle	335	100	piano	231	100
bowl	64	20	plant	240	100
car	197	100	radio	104	20
chair	889	100	range hood	115	100
cone	167	20	sink	128	20
cup	79	20	**sofa**	680	100
curtain	138	20	stairs	124	20
desk	200	86	stool	90	20
door	109	20	**table**	392	100
dresser	200	86	tent	163	20
flower pot	149	20	**toilet**	344	100
glass box	171	100	tv stand	267	100
guitar	155	100	vase	475	100
keyboard	145	20	wardrobe	87	20
lamp	124	20	xbox	103	20
			Total	9843	2468

5.3 Shapenet

ShapeNet (Chang et al. [2], 2015) is a richly-annotated, large-scale repository of shapes represented by 3D CAD models of objects. ShapeNet represents a collaborative effort between re-searchers at Princeton, Stanford, and TTIC (Toyota Technological Institute at Chicago). It consists in two subsets: ShapeNetCore and ShapeNetSem. ShapeNetCore subset contains single clean 3D models with manually verified category and alignment annotations. It has 55 object categories with 51, 209 unique 3D models; categories statistic are in Table 5.2 . This subset also covers the 12 object categories of the PASCAL 3D+, a popular computer vision 3D benchmark dataset. ShapeNetSem subset is a smaller, more densely annotated subset consisting of 12,000 models in a set of 270 categories. In addition to manually verified category labels and consistent alignments, these models are annotated with real-world dimensions, estimates of their material composition at the category level, and estimates of their total volume and weight. In our work we mostly used the ShapeNetCore (v1) subset. The 3D models data in ShapeNet Core have to be preprocessed before being fed to our neural network. Most of this preprocessing has been done using Python with packages such as NumPy [80] and Open3D [81].
For each ShapeNet Core 3D model:

1. Mesh clean up
2. Uniformly sample 2048 points from the mesh → Point Cloud Generation
3. Center and Scale to fit in unit box → Object ready.

5.4 ShapeNet Part

ShapeNet Parts [79] is a common benchmark dataset for Part Segmentation. Overall it contains 16,881 annotated models with the following split: 12137 training shapes, 1870 validation and 2874 shapes are for test. Shapes are from 16 different categories, depending on the category each shape can have from 2 to 6 parts. For each shape per-point part annotation is available. The Part Segmentation task consists in predicting for each point of a shape the associated part, for this reason this task can be seen as a dense classification problem in which we give a prediction for each point of each shape. Since possibile parts in a shape depend on the specific shape category we use category label (e.g. chair, airplane) for trimming irrelevant part predictions. Different categories have different vertex density distribution, in order to reduce density variability across categories we random sampled 2048 points from each shape.

Table 5.2. ShapeNetCore categories distribution. CAD models are not equally distributed between categories.

Name	Num	Name	Num	Name	Num
table	8443	jar	597	skateboard	152
car	7497	bottle	498	tower	133
chair	6778	bookshelf	466	camera	113
airplane	4045	laptop	460	basket	113
sofa	3173	knife	424	can	108
rifle	2373	train	389	pillow	96
lamp	2318	trash bin	343	mailbox	94
watercraft	1939	motorbike	337	dishwasher	93
bench	1816	pistol	307	rocket	85
loudspeaker	1618	file cabinet	298	bag	83
cabinet	1572	bed	254	birdhouse	73
display	1095	piano	239	earphone	73
telephone	1052	stove	218	microphone	67
bus	939	mug	214	remote	67
bathtub	857	bowl	186	keyboard	65
guitar	797	washer	169	bicycle	59
faucet	744	printer	166	cap	56
clock	655	helmet	162		
flowerpot	602	microwaves	152		
				Total	57386

5.5 ScanObject

ScanObject [62] is a real-world objects dataset extracted from two popular scene meshes datasets: SceneNN [82] and ScanNet [83]. The authors from a total of more than 1600 scenes selected a subset of 700 and then manually extracted objects from these scenes. The result is an *object dataset* with 2902 models categorised into 15 different categories. Each object is represented by a point cloud with 2048 points and its object category label. Normals, colors attributes, per-point part labels are also available, the latter being not currently available for all categories. This dataset, differently from the synthetic ModelNet and ShapeNet, aims to provide real-world 3d scans. Real-world 3D object scans are significantly different from synthetic models due to many factors including: presence of background and noise, object partiality caused by occlusion or incomplete scanning, different deformation

variants, non-uniform density *etc.* Despite the impressive state-of-the-art performance achieved by deep learning architectures on synthetic 3d data (ModelNet, ShapeNet), real world scans are way more complex and challenging for current learning methods. This dataset give us the opportunity to test the effectiveness of point cloud classification methods also in real-world setting. At the same time this new benchmark dataset raises a new issue, highlighting that current GDL methods poorly generalize to real-world domain, with *train on synthetic* → *test on real-world* scenario being not only the hardest in terms of overall test accuracy but also the one actually preventing intelligent systems from better reasoning real-world.
The dataset is randomly split into two subsets: 80% for training and 20% for test, these subsets contain objects from different scenes in order to avoid similar objects in the two. The authors provide different variants of their object scans: OBJ_ONLY, OBJ_BG, PB_T25, PB_T25_R, PB_T50_R and PB_T50_RS. Variants are listed in increasing difficulty order. The easiest one is OBJ_ONLY, containing only objects without background, this variant is the most similar to synthetic CAD data. Right after we have OBJ_BG in which objects are attached with background data, given a bounding box, all points in the box are extracted to form the object. Then we have the perturbations, each one introducing various degrees of background and partiality to objects. Suffixes T25, T50 denote random translation shifting the object bounding box up to 25% and 50% of its size from the box centroid along each world axis. Suffix R and S denotes rotation and scaling.

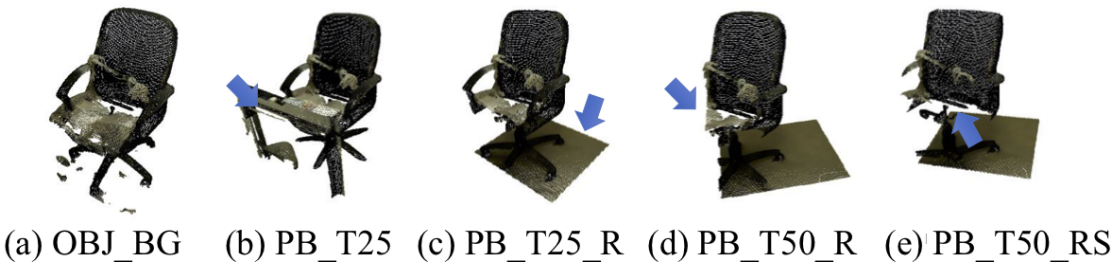

Figure 5.2. Example Chair from ScanObject dataset. In this figure it is shown the effect of different perturbation on a chair object, from the simplest OBJ_BG to the hardest PB_T50_RS. Image Source: [62]

Chapter 6

The Method

The main motivations behind this work are (1) coping with the lack of huge amounts of labeled data and (2) producing models with stronger generalization abilities, capable of dealing with real world challenges. While many approaches in literature have shown great results using supervised machine learning methods with 3D synthetic models (CADs) the main bottleneck of fully supervised methods still lies in the need of large annotated datasets for efficient learning. At the basis of our work there is the intuition that 3D point cloud understanding can still be extremely challenging even when supervised knowledge is provided. This becomes particularly true when moving from synthetic to real-world data and tasks. Infact there is no huge dataset of real world objects available and, furthermore, learning from real data is definitely not an easy task due to noise, presence of background, occlusions, data capture errors *etc.*
On top of that, models trained on synthetic data poorly generalize to real world data due to the strong gap between the two domains. We propose a novel way to exploit self-supervision for dealing with the lack of large quantities of annotated data and, at the same time, feeding the generalization gap across domains with different distributions.

6.1 Jigsaw Generalization

Self-Supervision can help in coping with lack of annotated data and lead to more robust machine learning models with increased generalization ability. Mehdi Noroozi and Paolo Favaro in [18] proposed a novel self-supervised task: the Jigsaw puzzle reassembly problem. The authors argue that by learning how to solve jigsaw puzzles the network learns that input objects are made of parts, what these parts are and how these are interconnected between them. Concept immediately translated in terms of features learned, more over they argued that these features can

be easily transferred to a multitude of different tasks like detection, classification, person re-identification and so on. In [55] the Jigsaw Puzzle task is re-formulated, demonstrating through experiments how this self-supervised task can be used for improving Image Object Recognition performance in a Domain Generalization setting. Previous deep learning jigsaw-puzzle solvers works formulated the jigsaw problem dealing with the separate tiles and then finding a way to recombine them. Although this type of strategy can be profitable for solving the task, on the other side, implies a tile-dedicated network architecture and eventually transferring the features learnt to a standard (non tile-specific) architecture. The approach proposed by [55] is quite different, a standard neural network architecture is used for both Obj. Classification and Jigsaw puzzle solving tasks in a completely multitask fashion. During training, a certain portion of the images in input to the network will be jigsawed. Since images in input must match in dimensionality, we can apply the same $n \cdot n$ tiles grid on each. Then, at random, selected images will be jigsawed by permutating the tiles in the grid. At training time, the network jointly learns to classify object categories and jigsaw permutation classes applied to images, in this way the jigsaw task is easily formulated as a classification problem. Jigsaw task is based only on intrinsic self-supervisory image signals, no extra labeling is required and, at the same time, allows capturing the natural invariances and regularities useful to bridge across different domains. Since the two task are jointly learnt by a standard neural network architecture, the features learnt by solving jigsaw puzzles are ready-to-use, with no need to transfer those features to another architecture.

6.2 Intuition

Due to their un-ordered nature, how to properly exploit local neighbours and at the same time taking into account the global structure of the 3D shape is quite challenging. As shown in [18, 31] a simple self-supervised task like reordering a 3D block puzzle can be solved by leveraging on the spatial co-location of shape parts and needs reliable knowledge on relative point positions both at global and local level. In this regard, while learning to solve a 3D puzzle we gain useful knowledge that can support recognition at different scales (whole object and parts). We formalize our learning model as a multi-task deep network where a main recognition task and the self-supervised puzzle task jointly learn a shared data representation. As we will see, the two tasks complement each other making the obtained features (1) more precise in case of large amount of labeled data, (2) more robust in case of scarce data, (3) easier to transfer for adaptation and (4) more reliable for out of domain generalization.

6.3 Method Formalization

Let us assume to observe data $S = \{(\boldsymbol{s}_i, \boldsymbol{u}_i)\}_{i=1}^{N}$ where each sample $\boldsymbol{s}_i$ is an order-invariant set of K points, $\boldsymbol{s}_i = \{p_1^i, p_2^i, \dots, p_K^i\}$. Each point $p_k^i \in \mathbb{R}^d$ is a vector describing its Euclidean coordinates (x, y, z) plus extra possible feature channels such as color, normal, *etc.* The corresponding label $\boldsymbol{u}_i$ depends on the specific task at hand. In case of *shape classification* it is a one-dimensional vector, so that each set of points is annotated with its belonging class identity $u_i \in \{1, \dots, C\}$ out of C categories. For *part segmentation*, instead $\boldsymbol{u}_i$ is a K-dimensional vector with each component $u_{ik} \in \{1, \dots, Q\}$ where Q is the number of object parts. In the following we will refer either to classification or part segmentation as our *main task* and we will describe how each of them can be jointly learned with the auxiliary self-supervised task of solving 3D puzzles. *Our multi-task model* can be described as the combination of two parametric non-linear functions: Φ_{θ_f,θ_m} and Ψ_{θ_f,θ_p}, where the subscripts of the parameters θ refer respectively to the feature extraction f, main task m, and puzzle portion p of our deep network. The *feature encoder* is shared between the two functions and is in charge of summarizing the local and global geometric information from the input point cloud to a richer latent space. The obtained representation enters the final fully connected part of the network that ends with the loss function $\mathcal{L}_m(\Phi(\boldsymbol{s}|\theta_f, \theta_m), \boldsymbol{u})$ measuring the error between the predicted and the true label. The auxiliary function Ψ deals with a variant of the original input point clouds. The samples are firstly scaled to unit cube before each axis is split into l equal lengths forming l^3 voxels which are labeled on the basis of their original position. This label is then used to annotate the internal points. Finally pairs of voxels are randomly selected and voxels in each pair are swapped between them, producing a new shuffled point cloud. We indicate with $Z = \{(\boldsymbol{z}_i, \boldsymbol{v}_i)\}_{i=1}^{N}$ the obtained *puzzled* samples where the voxel position label for each point is $v_{ik} \in \{1, \dots, l^3\}$. Once these new displaced data are encoded in the feature latent space, a second ending network branch focuses on solving the 3D puzzle problem by minimizing the auxiliary loss that measures the reordering error $\mathcal{L}_p(\Psi(\boldsymbol{z}|\theta_f, \theta_p), \boldsymbol{v})$ in terms of difference between the assigned voxel ID and the correct one per point.

Overall we train the network to obtain the optimal model through:

$$\arg\min_{\theta_f,\theta_m,\theta_p} \sum_{i=1}^{N} \Big\{ \mathcal{L}_m(\Phi(\boldsymbol{s}_i|\theta_f, \theta_m), \boldsymbol{u}_i) + \alpha \mathcal{L}_p(\Psi(\boldsymbol{z}_i|\theta_f, \theta_p), \boldsymbol{v}_i) \Big\} , \tag{6.1}$$

where both $\mathcal{L}_m$ and $\mathcal{L}_p$ are cross-entropy losses.

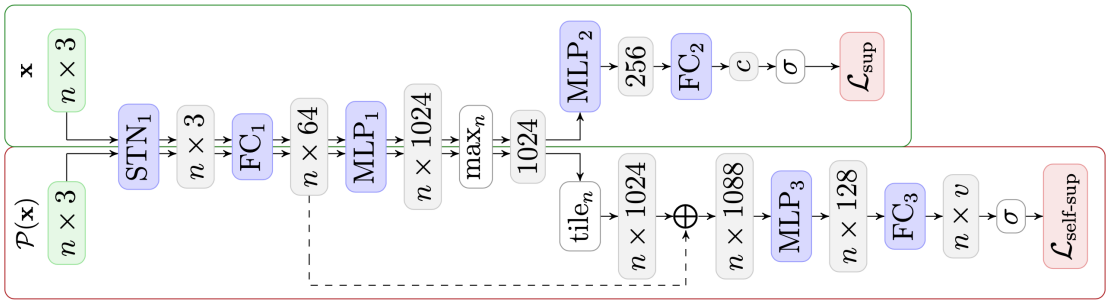

Figure 6.1. PN architecture used for shape classification. Color scheme: green = input data, blue = parametric layer, white = non-parametric layer, grey = output features, red = loss function.

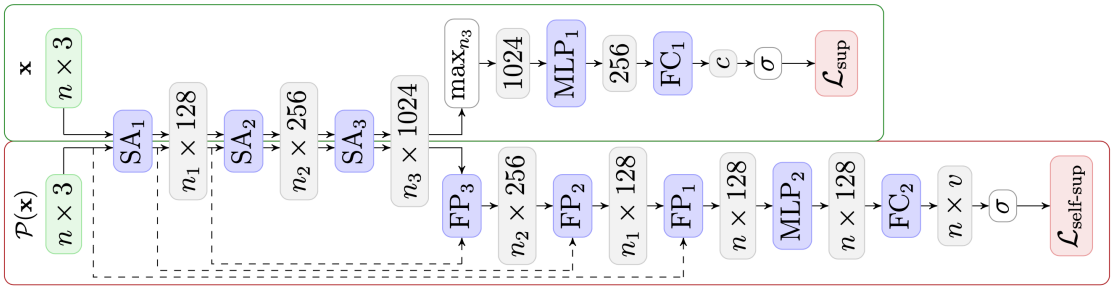

Figure 6.2. PN++ architecture used for shape classification. The color scheme is the same as Figure 6.1.

6.4 Implementation Choices

The described learning problem has two main hyper-parameters: one is α that weights the self-supervised loss, and the second, denoted as β, takes care of the data balancing in each sample batch during training. Specifically with β we indicate the percentage of the training batch composed of the original (*not-jigsawed*) samples. Both contribute to the trade off between the supervised and self-supervised tasks. Since we exploit self-supervision as an auxiliary objective we reasonably assign less importance to it with respect to the main task and set $\beta = \alpha = 0.6$ for all our analysis. We reserved some variations for β only when investigating the small-sample scenarios (few shot and semi-supervised). A further parameter is the quantization step of the space to define the puzzle parts: we set $l = 3$ and considered its reduction to $l = 2$ only in the single domain setting on complex real-world data. Finally, in the puzzle creation phase we always kept one single permuted copy of each sample. To realize our model we built over two well known and reliable architectures: PointNet and PointNet++ by extending their structure

with the inclusion of a new ending branch dedicated to 3D puzzle resolution. By investigating both of them as starting point we can highlight the different

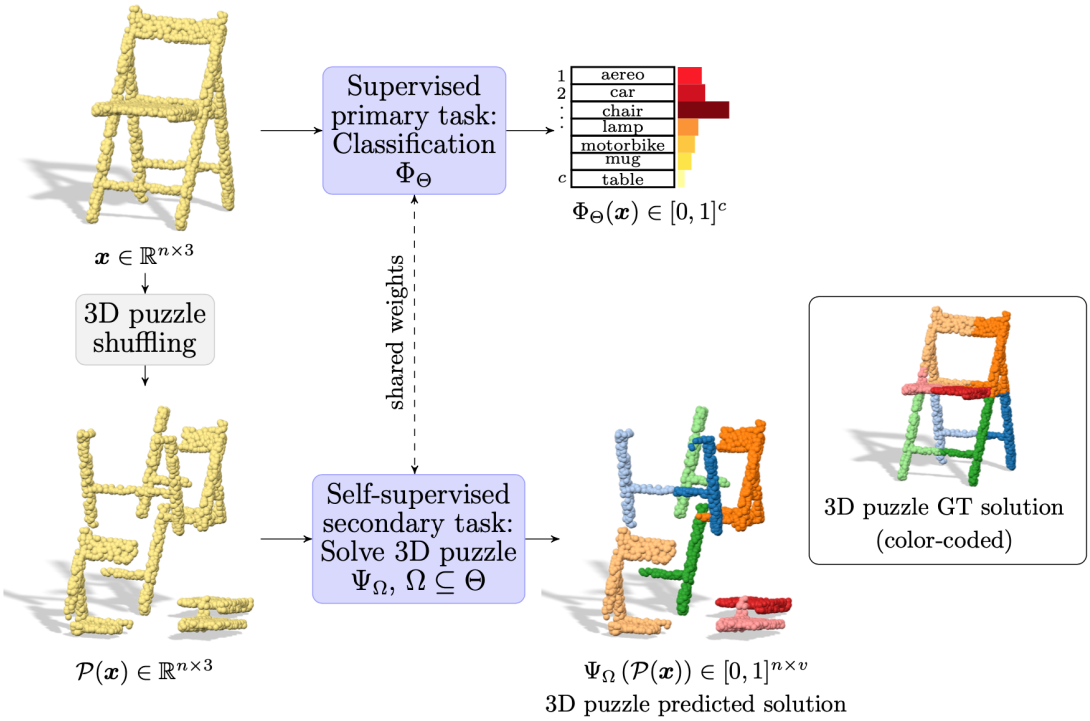

Figure 6.3. Illustration of our multitask architecture. Two tasks are jointly solved: Object Classification and Jigsaw puzzle solving. Weights are shared between the two tasks.

effect of the context information learned with the puzzles to different ways of dealing with point clouds. Indeed the first architecture basically learn on each point independently and only accumulates the final features, while the second follows a multi-scale strategy with a heuristic point grouping at separate layers. In both case, the main feature encoder is shared between the two tasks, only few task-specific layers are added at the head of the network. Our proposed architectures, inspired by [66] and [55], perfectly fits into the context of hard parameter sharing multitask learning. Figure 6.1 and 6.2 illustrate the corresponding architectures for our multi-task approach.

Out of the l^3 possible input point cloud permutations we offline selected and stored a random set of 31 permutation classes P, we kept the same set throughout all our experiments. This set includes the permutation zero ρ_0 which corresponds to the not jigsawed case. Voxelization is fundamental in our pipeline since we define the Jigsaw puzzle as a rigid swap between voxels in selected pairs. While the

jigsaw branch operates on the whole set of input point clouds (jigsawed or not) we let the main task branch working only on not jigsawed pcs. This is realized by doing back-propagation on main task specific layers only for not jigsawed indices. Since working in a multitask learning fashion, with the feature encoder shared between the two tasks, features learnt by Jigsaw puzzle solving will be immediately available and exploitable by the primary task with no need to transfer the jigsaw learnt features to another standard architecture. Furthermore, we claim that the Jigsaw task has a regularization effect on the main task, natural invariances and regularities learnt by solving the Jigsaw puzzle can be useful to get better the main task.

Our approach is synthesised in 6.3. The network is trained for jointly solve two tasks: Object Classification (main task) and Jigsaw puzzle solving (secondary task). The network not only has to predict the object category $u_i \in \{1, \dots, C\}$ out of C categories of the observed point cloud but has also to solve jigsaw puzzles. Given a shuffled 3d shape the puzzle solving problem has been formulated as a segmentation (or *voxel prediction*) task, the network assign to each of the input points a label $v_{ik} \in \{1, \dots, l^3\}$ where l^3 is the total number of voxels. For better visualization, in Figure 6.3, voxels are represented with colors, each of them is associated to a label v_{ik}.

Jigsaw Solving (Visualization): In Figure 6.4 we show the 3D puzzle task getting progressively solved by our model during training. At each epoch we tested the performance of the jigsaw puzzle solver on few shapes. Out of the 100 total epochs we selected few relevant ones with the intent of visualizing the jigsaw predictions. We remind the reader that the Jigsaw puzzle task is formulated as per-point voxel prediction. In Figure 6.4 we associate to each voxel index a unique color. We expect that in initial epochs visualization colors get confused, this because the Jigsaw task, not sufficiently trained, will output wrong random per-point original voxel prediction. After a certain number of epochs, the Jigsaw Puzzle task cost function converge, meaning that the Jigsaw predictions from that moment should be valuable, points get uniformly colored accorded to their original (before puzzling) position. In first and second row the shuffled part of the lamp and of the car at the beginning (epoch 0) are almost uniformly wrong. Then, from epoch 10 going on, the predicted voxels starts to get well separated. Third row shows that annotating the correct original voxel appears particularly difficult for this table due to the presence of multiple similar subparts. In last row, airplane case, seems easy to have coherent predictions per voxel but the voxel identity is initially mistaken and progressively corrected only in the following epochs.

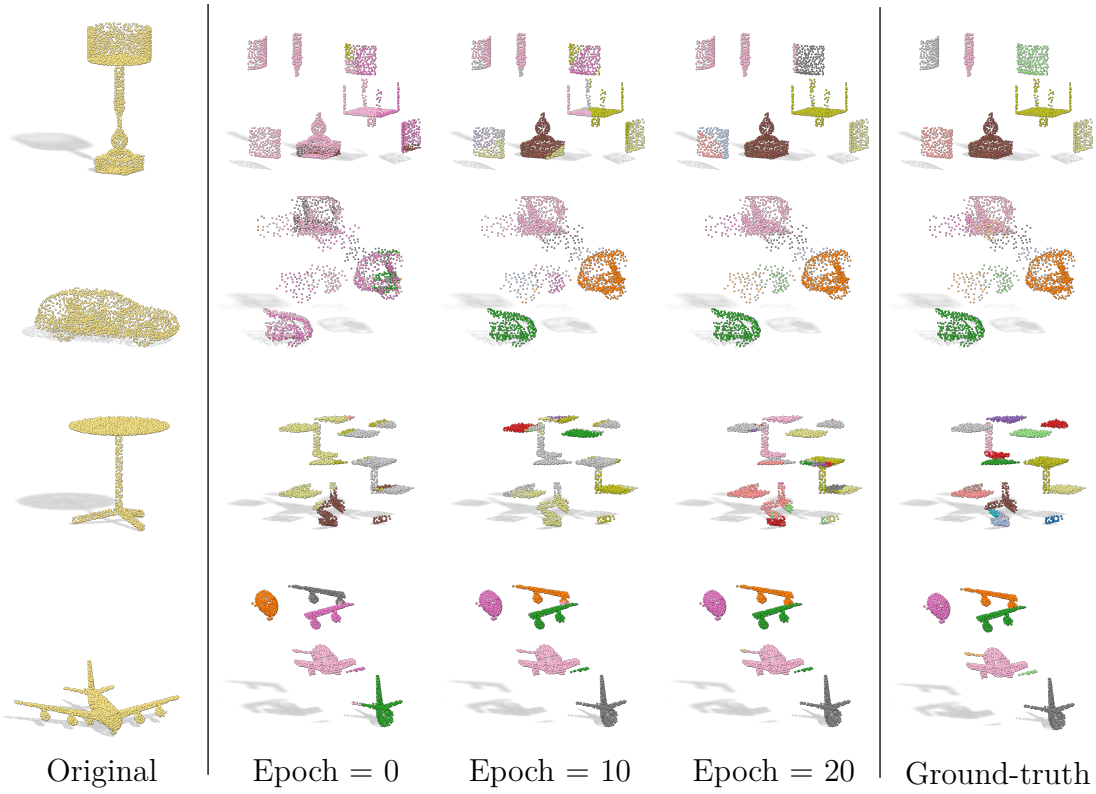

Figure 6.4. Visualization of 3D puzzle progressively solved by our model. First and second row: the shuffled part of the lamp and of the car at the beginning are almost uniformly wrong. The different voxels are then separately identified during training. Third row: annotating the correct original voxel appears particularly difficult for this table case due to the presence of multiple similar subparts. Fourth row: for the airplane it seems easy to have coherent predictions per voxel but the voxel identity is initially mistaken and progressively corrected only in the following epochs.

Chapter 7

Experiments

In this chapter we describe the experimental part of this work. In order to validate the value of our approach we tested it in many acknowledged computer vision settings including Domain Adaptation and the more challenging Domain Generalization. In the first part of this chapeter we describe experiments with Object Classification as primary task while in the second we discuss experiments with Part Segmentation.
Results, provided in tables, will help the reader in understanding the contributions of our approach with respect to the previous state-of-the-art methods.

7.1 Preface

Working on raw 3D data is pretty computational expensive. For this reason and also encouraged by the impressive results obtained in the 2D field with CNN, computer vision community has first tried to deal with the 3D by reducing it to a 2D problem.
According to this approach 3D data were reduced into multiple 2D images and a multiview-based CNN was trained to solve the object classification problem in the 2D field. Recent works proposed a new approach, directly learning from unordered point sets (also known as point clouds) without any need to project or map 3D (or higher dimensionality) data to lower dimensionality space. PointNet [5] is a novel type of neural network consuming point clouds which respects the permutation invariance property of points in input, for sure the pioneer in this direction. In our experiments we also follow this direction, we always work directly on point clouds and we mainly use PointNet and PointNet++ as our backbones. Most of the experiments performed in this work will be reported and explained in this chapter, through all experiments we followed specific settings commonly acknowledged by

the machine learning community. All these settings can be divided into two categories which are Single Domain and Across Domains. With *Single Domain* we intend that source domain and destination (or target) domain are from the same distribution. There is no domain shift between the source and target. This is usually achieved by training on a split of a dataset and then testing on a different and never-seen split of the same dataset. In case of *Across Domains* source distribution is different from target distribution. This is a way more challenging setting because the machine learning model must learn domain agnostic features from source domain and then exploit those features on a target domain with a different distribution.

Training details Throughout all the experiments we used PyTorch as our machine learning framework. The training is conduced on a NVIDIA Titan X GPU. Models are always trained from scratch, training parameters as batch size, learning rate and the scheduler parameters are chosen according to the specific backbone used, the dataset and the specific task (obj. classification, part segmentation etc.). In general for PointNet we used a batch size of 64, Adam optimizer with initial learning rate 0.001, decreased by a factor of 4 every 20 epochs. For PointNet++ we used batch size of 64, SGD optimizer with momentum 0.9, initial learning rate 0.01, decreased by a factor of 2 every 20 epochs. Other parameters are the one related to the Jigsaw selfsupervised task. Data augmentation is performed only on ScanObjectNN by following verbatim the procedure proposed by [5], *i.e.* random vertex jittering drawn from $\mathcal{N}(0, 0.01)$, and random rotation around the shape elongation axis.

7.2 Settings

We consider different experimental settings involving a source dataset S divided into S_{train} and S_{test} parts, a possible extra set of unlabeled data from a different source domain S' and an unlabeled target domain T, different from both S and S'. In the following we refer to the different settings as

Single Domain the whole set of annotated samples from S_{train} are available for supervised learning. Finally we test on the portion S_{test} of the same original dataset.

Few-Shot under this name we investigate the case of limited training data samples. We reduce at different percentage scales the cardinality of S_{train} and we evaluate on S_{test}. Regardless of this reduction we follow the same batch learning procedure of the previous setting as well as the same final evaluation protocol.

This setting aim to test model robustness when decreasing the quantity of supervised data. Our expectation is that by jointly learning jigsaw puzzle task valuable features will be learnt, those features should positively influence the main segmentation or classification task leading to a less label greedy method.

Semi-Supervised in this case S_{train} is divided into two complementary $\lambda\%$ supervised and $(1-\lambda)\%$ unsupervised parts.The first is provided as input to the main task of our network, while the second goes to the self-supervised task. Finally we test on S_{test}. The difference with the previous few-shot setting is in the fact that held out portion of S_{train} is taken as unlabeled data, fed to our jigsaw puzzle solver task.

Transfer Learning (TL) here besides the annotated data from S, a further set of unlabeled samples from a different domain S' is available at training time. In this case when running the multi-task approach we feed the self-supervised task with the extra unlabeled samples while the supervised data is only used for the main task. The final evaluation is performed on S_{test}.

Domain Generalization (DG) this setting is analogous to the Single Domain one and differs only in the evaluation phase where the performance is finally computed on a new target collection T belonging to a different domain with respect to the supervised data S on which the model is trained.

Domain Adaptation (DA) here we consider available at training time both the supervised data S and the unsupervised target data T, belonging to two different domains. Only the target is given as input to the self-supervised part of our multi-task method, while the main task is learned on S. Finally we transductively evaluate the model on the same data T as standard practice in domain adaptation.

7.3 Object Classification Results

First part of our experimental analysis is dedicated to the main task of shape classification. The goal is to predict the object category of the observed point cloud out of a set of C classes. We evaluate the performance in terms of overall accuracy.

7.3.1 Single Domain

In the first experiment we evaluate the performance of our multitask approach in the most classical scenario when a single domain is available and present the results

in Table 7.1. Since we are in a single domain setting source domain and target domain distribution exactly match, with no shift between them. In this experiment we train our network to jointly solve two tasks: Object Classification primary task and Jigen self-supervised secondary task. We expect that by jointly solving the Jigsaw puzzle self-supervised task our model will be able to learn semantically relevant features concerning object parts, their spatial correlation and arrangement. This secondary task not only helps the network to learn the concepts of spatial correlation but also acts a regularizer for the primary Object Classification task. We consider as testbed ModelNet40 (C=40) and ScanObjectNN (C=16), respectively a synthetic and a real-world dataset. We observe that in both cases our multitask approach consistently outperform over its baseline regardless of the specific considered backbone. These results indicate that by simply solving the auxiliary self-supervised task the learned representation is better able to capture the object semantic provides further discriminative information to the final classifier. The advantage appears particularly evident on the real world dataset ScanObjectNN with a gain of about 3 percentage points (pp) for the difficult `PB_T50_RS` with and without background. This further confirm the effectiveness of the proposed approach to tackle the most challenging real world scenarios.

7.3.2 Transfer Learning

We also perform TL experiments considering the availability of an extra source S' of unsupervised knowledge. In particular, with the aim of maintaining a good balance between S' and the annotated S we used a random subset of 5k samples from the ShapeNet (S') dataset for the experiments on ModelNet40 (S) and a random sample of 4k samples from training set of ModelNet40 (S') for the experiments on ScanObjectNN (S). The first TL case allow us to study knowledge transfer among two different synthetic domains while the second correspond to knowledge transfer from synthetic to real point clouds. The results are in the last row of Table 7.1. Overall we observe again a further improvement up to 1 pp with respect to the previous results without transfer, with the only exception of ModelNet40 and `OBJ_ONLY` for PN++. Here the extra synthetic information does not seem to actually provide useful information and also degrades the advantage provided by the self-supervised task on the original source S. However this is not true for PN, indicating that also the backbone network has a role in the specific transfer process. We highlight that, although previous work have discussed TL in combination with self-supervised tasks, the used setting differ from ours. In [31], the whole ShapeNet dataset is used to train a model for 3D puzzle solution and the obtained weights are transferred to initialize supervised learning on ModelNet40. The obtained accuracy is 92.4%, while the baseline with random initialization is 92.2%. Considering the amount of extra unsupervised data used there and that

the used learning architecture is the powerful DGCNN [11], our 92.10% obtained without extra information appears upstanding. Another interesting comparison can be done with the multi-task method recently presented [62] which performs classification and segmentation jointly as a possible strategy to deal with real world point clouds. Although effective, this approach doubles the need of data annotation producing an accuracy of 80.2% on `PB_T50_RS_BG` over a baseline of 77.9%. With a matching baseline, our result is 79.08% without using any extra labeled annotation and confirm the effectiveness of the auxiliary self-supervised task.

7.3.3 Limited Annotated Data

As previously said one of the main bottleneck for efficient 3D deep learning is the need for huge datasets largely annotated. In this experiment we demonstrate that self-supervised learning can be a valuable ally in coping with this issue. We show how the performance of trained models vary widely depending on the quantity of training data and how self-supervision makes our learned models more robust to limited quantity of annotated training data. We focused on ModelNet40 for experiments in the few shot and semi-supervised setting. The results in Table 7.2 show the performance obtained when the amount of labeled training data reduces up to only 20% of the original amount. We can observe that, despite the overall drop in performance when reducing the quantity of training data, our multi-task approach in the few shot setting tends to maintain its advantage with respect to the baseline. When considering also the unlabeled data for the semi-supervised setting, we observe a further small increase in performance at least with PN.

Shape Classification						
Method		ModelNet40	ScanObjectNN			
			`OBJ_ONLY`	`OBJ_BG`	`PB_T50_RS`	`PB_T50_RS_BG`
PN	baseline	88.65	75.22	70.22	71.37	62.56
	our multitask	89.71	75.04	**71.26**	73.39	65.20
	our multitask TL	**90.72**	**77.45**	**71.26**	**73.49**	**65.61**
PN++	baseline	91.93	84.17	83.99	78.66	77.90
	our multitask	**92.10**	**85.89**	83.13	79.22	78.00
	our multitask TL	91.58	84.68	**84.33**	**80.46**	**79.08**

Table 7.1. Shape classification accuracy (%) of our multi-task approach with respect to the main classification baseline implemented on two different backbone architectures (PN [5], PN++ [7]). In the Transfer Learning (TL) setting, extra unsupervised data sources are integrated in the learning process as input to the self-supervised task (ShapeNet for ModelNet40 and ModelNet40 for ScanObject). The suffix `BG` indicates that the point clouds contains background vertices as well.

Shape Classification with limited annotated data					
Method		ModelNet40			
		20%	40%	60%	100%
PN	baseline	82.94	85.49	87.11	88.65
	our mult. (few-shot)	82.09	87.03	88.13	**89.71**
	our mult. (semi-sup.)	**83.06**	**87.27**	**88.57**	-
PN++	baseline	85.37	88.25	89.63	91.93
	our mult. (few-shot)	**86.35**	**89.59**	89.18	**92.10**
	our mult. (semi-sup.)	86.02	88.41	**89.83**	-

Table 7.2. Shape classification accuracy (%) of our multi-task approach when the training set contains a limited amount of annotated data.

7.3.4 Domain Adaptation and Generalization

When training and test data are drawn from two very different distributions the learned model usually fails to generalize. Being able to maintain a good performance in this challenging setting is crucial in all the cases in which accessing to annotated data of the target domain is not possible.
In our experiments it's particularly evident that when shifting from synthetic source domain (as ModelNet40) to real world target domain (as ScanObjectNN) deep learning models fails to generalize. Since annotated real world data are not largely available we still want to exploit synthetic data as source, we propose to reduce the domain gap by exploiting our multitask approach. We consider the DG setting when training on ModelNet40 and testing on ScanObjectNN and report results in Table 7.3. Our multitask approach fully learned only on synthetic data shows a significant better performance with respect to the baseline with improvements up to 6 and 8 pp in the `OBJ_BG` and with a still relevant gain of 2 and 4 pp in the most challenging `PB_T50_RS_BG`, respectively with PN and PN++. We also consider the inverse generalization direction, using real world data as source and synthetic data as target, from `PB_T50_RS_BG` to ModelNet40. Here with PN we observe a drop in performance, while PN++ shows the opposite behaviour still presenting a large gain. To complete the analysis we also investigated whether our multi-task approach could close the domain gap between the considered domains. The results in the DA setting provide a positive answer showing a further increase in performance over the DG results with only few exceptions with PN. An overall look to the behaviour several recent point cloud networks (top part of Table 7.3) indicates that with our multitask approach we are establishing the new state of the art for classification on real world data from synthetic training. Even in the opposite learning direction from real to synthetic, our multitask shows promising results and the ability to adapt provides it with a further way to improve over

existing references.

Domain Generalization and Adaptation						
Method		ModelNet40 →				PB_T50_RS_BG →
		OBJ_ONLY	OBJ_BG	PB_T50_RS	PB_T50_RS_BG	ModelNet40
3DmFV [10]		30.90	24.00	24.90	16.40	51.50
SpiderCNN [9]		44.20	42.10	30.90	22.20	46.60
DGCNN [11]		49.30	46.70	36.80	27.20	54.70
PointCNN [8]		32.20	29.50	24.60	19.20	49.20
PN	baseline [62]	42.30	41.10	31.10	23.20	50.90
	our baseline	54.74	43.58	44.96	34.25	47.43
	our multitask DG	54.53	**49.68**	45.22	36.28	39.30
	our multitask DA	**58.53**	47.58	**46.70**	34.46	**51.54**
PN++	baseline [62]	43.60	37.70	32.00	22.90	47.40
	our baseline	52.84	44.00	44.83	34.29	47.66
	our multitask DG	56.84	52.42	52.84	38.65	52.88
	our multitask DA	**60.84**	**53.89**	**54.66**	**39.63**	**56.07**

Table 7.3. Shape classification accuracy (%) of our multi-task when training and testing is done on different domains (DG). If the unlabeled target data is provided at training time (DA), our multitask is able to adapt and reduce the domain gap.

7.4 Part Segmentation Results

The second set of experiments is dedicated to part segmentation task. Given a 3D point cloud represented by an unordered set of vertices, part segmentation goal is to assign a part label to each vertex. Typically, part segmentation is modelled as a point-wise classification problem where each shape part is represented by an integer label in $\{1, \ldots, P\}$. Following [5], the quality of the predicted part segmentation is evaluated in terms of the mean Intersection-over-Union (mIoU) metric. The mIoU of a shape with P parts is defined as the average over P parts of the IoU between ground-truth and predictions of each part. The mIoU of a category (e.g. chair) is defined as the average of the mIoUs of all the point clouds referring to that category.

Training details: throughout all the part segmentation experiments we used the PointNet Part Segmentation backbone from [5]. We slightly modified the network architecture introducing a branch for jointly solving the Jigsaw self-supervised task, this branch shares most of the inital network layers with the main segmentation one. Our modifications does not increase the original segmentation branch capacity. A batch size of 64 has been used, further more the chosen optimizer is Adam with an initial learning of rate 0.001, decreased by a factor of 2 every 20

epochs. Data augmentation is applied only on ScanObjectNN exactly as in our classification experiments.

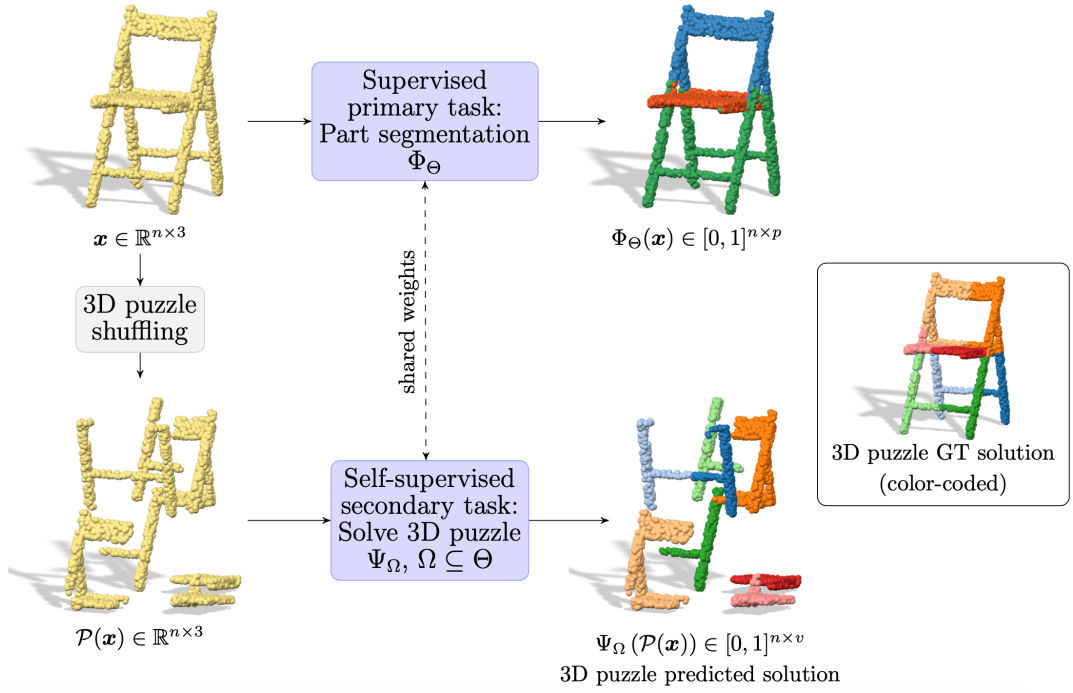

Figure 7.1. Illustration of our multitask architecture for Part Segmentation. Two tasks are jointly solved: Part Segmentation and Jigsaw puzzle solving. Weights are shared between the two tasks.

7.4.1 Few Shot and Semi-Supervised Results

In this experiment we wanted to study how limited amount of annotated data affect Part Segmentation perfomance. Following [33] we randomly sample 1% and 5% of the *ShapeNet Part* train set to evaluate our multitask approach in a semi-supervised setting. Since we embrace a semi-supervised approach we will use only the 1% (or 5%) of the overall training ShapeNet Part dataset as supervised training source, the remaining 99% (or 95%) will be used as self-supervised data, fed to our Jigsaw puzzle solver task. The results in Table 7.4 indicates that our multitask, although not improving over the baseline in the few-shot setting, in the semi-supervised setting outperforms the current state of the art in the 1% case and practically matches it in the 5% case. It is interesting to underline that also our best competitor CCD operates exactly in our same semi-supervised setting and is a multitask approach that combines clustering and reconstruction with a

self-supervised classification obtained by learning on the clustering auto-defined labels.

Part Segmentation: mIoU in case 1% of supervision		
Method	1%	5%
SO-Net [26]	64.0	69.0
PointCapsNet [27]	67.0	70.0
CCD [33]	68.2	**77.7**
baseline	64.52	75.75
our multitask (few-shot)	64.49	75.07
our multitask (semi-sup)	**72.95**	77.42

Table 7.4. Intersection-over-Union (mIoU) accuracy for ShapeNetPart segmentation.

For a more in-depth analysis of our results we also plotted some visualizations out of our 1% part segmentation experiment in Figure 7.2 and 7.4. In Figures 7.2 and 7.3, our multitask approach seems to allow a better recognition of the chair armrests. Indeed the position of these relative small parts of the chair may be better learned thanks to the puzzle solution task. A similar consideration may be done for the lamp basis in Figures 7.4 and 7.5 .

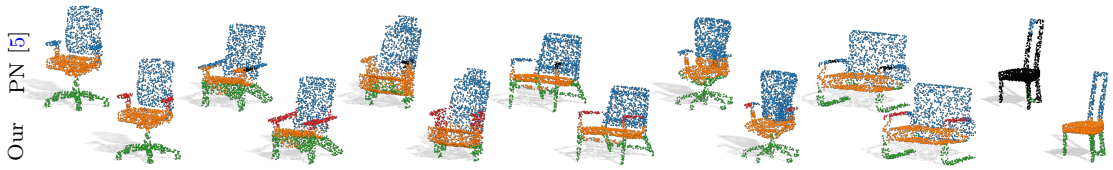

Figure 7.2. Part annotation of the chairs when only 1% of training data are available. On the top left we show the predictions obtained by the baseline, on the bottom right the predictions obtained by our approach. On the last column we show the worst case for the baseline. Black points denotes predictions whose maximum value was not a chair part.

7.4.2 Single Domain and Transfer Learning on Real World data

We wanted to evaluate the performance of our multitask approach also on the most challenging ScanObjectNN real world dataset. At the time of the work this dataset was not publicly available, available only to individuals who requested to download for academic purposes. ScanObjectNN kindly gave us access to real-world 'chairs'

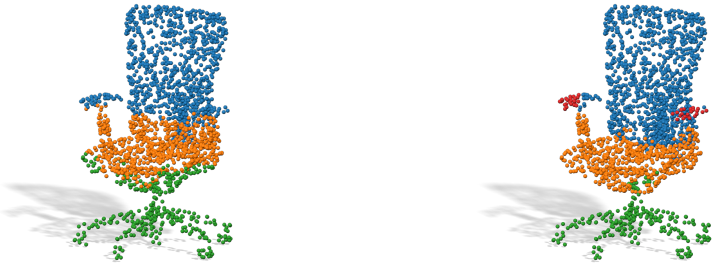

Figure 7.3. Part annotation of the chairs when only 1% of training data are available. On the left we show the predictions obtained by the baseline, on the right the predictions obtained by our approach. Our approach reveals to be more robust in case of limited data producing better per-point part predictions, in this example we highlight that the baseline not only produces a worse prediction but also completely miss chair's arms (red colored).

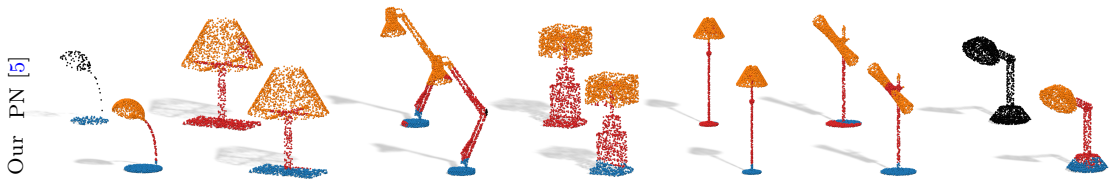

Figure 7.4. Same visualization of Figure 7.2 but for the lamp category.

Figure 7.5. Same visualization of Figure 7.3 but for the lamp category. The baseline (left side) miss lamp basis parts, our approach (right side) producing a better prediction in case of limited training data.

part-annotated dataset. We start by considering the class chair from two of the subsets of ScanObjectNN and running our multitask for part segmentation on

them. We present the results in Table 7.5 adopting for this case the same evaluation metric used in [62]. Both the per-class average and the overall accuracy indicate that in this case the introduction of self-supervision in the learning process does not have a visible effect and in the more difficult PB_T50_R_BG subset it may also slightly decrease the performance. We also extended the analysis to the TL setting including as extra source of knowledge the unlabeled samples of class chair from ModelNet40: in this case the lack of annotation refers to the missing part label for each point. In this setting our multitask approach presents a small advantage, more visible in the easier OBJ_BG, confirming its effectiveness.

Part Segmentation								
Dataset	Method	Bg	Seat	Back	Base	Arm	Avg. Acc.	Overall Acc.
OBJ_BG	our baseline	65.14	**87.88**	**89.73**	67.16	58.97	73.02	81.62
	our multitask	64.97	87.46	86.27	68.96	57.54	73.04	81.67
	our multitask TL	**69.43**	86.59	88.71	**72.70**	**61.37**	**75.76**	**82.60**
PB_T50_R_BG	our baseline	82.06	83.71	75.30	54.75	35.11	66.19	81.13
	our multitask	82.02	83.50	77.80	51.53	25.50	64.07	81.31
	our multitask TL	**83.08**	81.87	**79.06**	50.71	30.40	64.99	**81.82**

Table 7.5. Per part average accuaracy (%) and overall accuracy (%) of part segmentation of chairs in two variants of ScanObjectNN, in both chosen variants background points are present.

7.4.3 Domain Generalization and Adaptation

As done for shape classification we complete the cross-domain analysis also for the part segmentation task by running experiments from synthetic to real ScanObject chairs. Specifically we used the same datasets already described in the paper ShapeNetPart as source and ScanObjectNN as target. We highlight that ScanObjectNN has some part annotation issue confirmed by the authors through personal communications, thus we prefer to use only the OBJ_BG provided subdomain, neglecting the background which is absent in ShapenNetPart. Table 7.6 collects the results for this setting confirming also in this case the advantage of our multitask approach over the supervised learning baseline. As soon as a real world part-annotated dataset will be publicly available we will repeat the Domain Generalization and Adaptation experiments on a bigger set of object categories.

Part Segmentation - Domain Generalization and Adaptation					
ShapeNetPart → `OBJ_BG`					
Method	Seat	Back	Base	Arm	Parts Avg.
our baseline	67.85	45.60	84.89	14.87	53.30
our DG	**71.80**	42.61	84.57	**21.48**	55.11
our DA	65.70	**49.11**	**85.91**	21.40	**55.53**

Table 7.6. Per part and average accuracy (%) of chair segmentation. We are adopting the same metric used in Table 4 of the main paper. All the results are average values over three runs.

Chapter 8

Conclusions

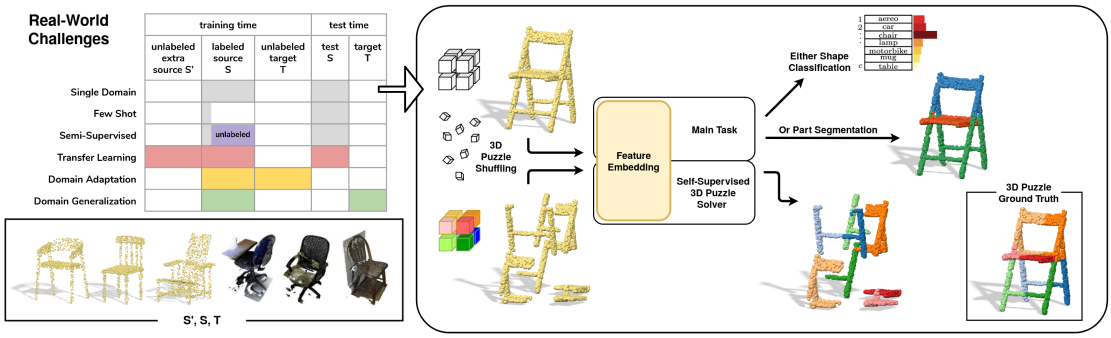

Figure 8.1. Illustration of our multitask framework. We accept the 3d real–world challenges proposing our 'recipe' to deal with, our method takes advantage of self-supervision by solving jigsaw puzzles on shuffled input signals. We propose a complete framework for Classification and Part Segmentation, achieving state-of-the-art performance in case of limited amount of training data and generalization from synthetic to real world domain.

In the introduction, we presented the *3d challenges* that we briefly sum up with: (1) reducing the need for large annotated training data for efficient learning, (2) need for learning methods with good generalization ability on real-world data. We concentrated our analysis on a specific type of *Geometric Data* [6] represented by 3d point clouds, unordered sets of points lacking of Euclidean structure. Point cloud data annotation process is expensive and time-consuming, depending also on the annotation type required by the specific learning task. Recently multiple synthetic point cloud and mesh datasets came up: ModelNet [1] and ShapeNet [2] have been the forerunners, pushing for effective learning methods consuming this type of data. However, synthetic point clouds are missing the artefacts usually existing

in real-world point cloud scans acquired with 3D LiDAR or others 3d sensors. As a result, the performance of models trained on synthetic degrade when tested on real-world due to the domain shift between the two domains. Despite this, from our point of view synthetic data are still crucial since huge labeled real-word 3d datasets for effective learning are still lacking. In this context, several recent works exploit self-supervision also in the 3d domain [32, 61, 33, 31]. The notion of self-supervision appears extremely relevant for 3D structures, both to capture internal local information at a good neighborhood resolution before combining it with global shape knowledge, and to get information on context and surfaces. While most of related works in 3d domain take advantage of self-supervision in a two-phase learning fashion [31, 32], we propose a different approach consisting in a *multi-task* end-to-end learning architecture. Our multi-task architecture, built on top of the reliable PointNet [5] and PointNet++ [7], jointly learns two task: the primary task (classification or segmentation) and the jigsaw puzzle secondary task. The intuition is that solving the puzzle self-supervised task allows capturing the natural invariances and regularities useful to bridge across different domains. This intuition is brightly confirmed by our experiments. We tested the performance of our approach with Object Classification and Part Segmentation primary tasks. In both cases we placed great emphasis on tackling the *real-world* challenges, testing our multitask architecture not only in Single Domain setting but also in Domain Adaptation (*DA*) and Domain Generalization (*DG*) settings. In case of Object Classification we mostly used two datasets: ModelNet40 [1] which is a very popular synthetic dataset for Obj. Classification and ScanObjectNN [62], a real-world object scans dataset providing several splits with increasing difficulty (in the sense of background, noise, presence of occlusions *etc.*). Differently, for Part Segmentation experiments we needed further annotated data with per-point part labels, our choice fell on ShapeNet Part [79] for synthetic and, once again, ScanObjectNN for real-world annotated scans.

Results in Single Domain setting indicate that by simply solving the auxiliary self-supervised task the learned representation is better able to capture the object semantic providing further discriminative information to the final classifier, the advantage is particularly evident on real-world data, see Table 7.1. Our multi-task has proven to be robust to limited quantity of training data, in semi-supervised setting we outperform previous state-of-the-art method [33] for Part Segmentation with limited annotated data, *i.e.* using only 1% of annotated data, see Table 7.4. In DA and DG cases we discussed the following domain shifts case:

(1) Synthetic $\rightarrow$ Real-World

(2) Synthetic $\leftarrow$ Real-World

the first (1) being the one actually preventing intelligent systems from reasoning about 3d space more effectively. In Table 7.3, we confirm the value of our approach also for DA and DG setting, establishing with our multitask the new state-of-the-art for classification on real-world data from synthetic training. Even on the opposite direction, from real to synthetic, our multitask shows promising results and the ability to adapt provides it with a further way to improve over existing references. The remarkable results obtained by our multitask highlight the importance of methods able to deal jointly with labeled and unlabeled data possibly coming from different domains. Also the spread of different 3d sensors, easily deployable on intelligent systems, suggests a future huge availability of unlabeled 3d real-world scans. In this direction ours multi-task approach (Figure 8) combines supervised and self-supervised learning with compelling results in all the considered scenarios over several real world and synthetic datasets. We see this work as a first exciting step towards a new family of methods better able to generalize and adapt to novel testing conditions for 3D point clouds. Our choice of the specific self-supervised task of solving 3D puzzle is indeed just one of the many possible that deserve attention for *future work*.

Bibliography

[1] Zhirong Wu et al. *3D ShapeNets: A Deep Representation for Volumetric Shapes*. 2014. arXiv: 1406.5670 [cs.CV].

[2] Angel X. Chang et al. «ShapeNet: An Information-Rich 3D Model Repository». In: *CoRR* (2015).

[3] Danica Kragic and Markus Vincze. «Vision for Robotics». In: *Foundations and Trends in Robotics* 1 (2009), pp. 1–78. DOI: 10.1561/2300000001.

[4] Arsalan Mousavian, Clemens Eppner, and Dieter Fox. «6-DOF GraspNet: Variational Grasp Generation for Object Manipulation». In: *CoRR* (2019). URL: http://arxiv.org/abs/1905.10520.

[5] Charles R Qi et al. «PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation». In: *arXiv preprint arXiv:1612.00593* (2016).

[6] Michael M. Bronstein et al. «Geometric Deep Learning: Going beyond Euclidean data». In: *IEEE Signal Processing Magazine* 34.4 (2017), pp. 18–42. ISSN: 1053-5888. DOI: 10.1109/msp.2017.2693418. URL: http://dx.doi.org/10.1109/MSP.2017.2693418.

[7] Charles R Qi et al. «PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space». In: *arXiv preprint arXiv:1706.02413* (2017).

[8] Yangyan Li et al. «PointCNN: Convolution On X-Transformed Points». In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio et al. Curran Associates, Inc., 2018, pp. 820–830. URL: http://papers.nips.cc/paper/7362-pointcnn-convolution-on-x-transformed-points.pdf.

[9] Yifan Xu et al. «SpiderCNN: Deep Learning on Point Sets with Parameterized Convolutional Filters». In: *arXiv preprint arXiv:1803.11527* (2018).

[10] Yizhak Ben-Shabat, Michael Lindenbaum, and Anath Fischer. «3DmFV: Three-Dimensional Point Cloud Classification in Real-Time Using Convolutional Neural Networks». In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3145–3152.

[11] Yue Wang et al. «Dynamic Graph CNN for Learning on Point Clouds». In: *ACM Transactions on Graphics (TOG)* (2019).

[12] Chu Wang, Babak Samari, and Kaleem Siddiqi. «Local Spectral Graph Convolution for Point Set Feature Learning». In: *ECCV*. 2018.

[13] Li Yi et al. «SyncSpecCNN: Synchronized Spectral CNN for 3D Shape Segmentation». In: *CVPR*. 2017.

[14] Deepak Pathak et al. «Context Encoders: Feature Learning by Inpainting». In: *CVPR*. 2016.

[15] Richard Zhang, Phillip Isola, and Alexei A Efros. «Colorful Image Colorization». In: *ECCV*. 2016.

[16] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. «Colorization as a Proxy Task for Visual Understanding». In: *CVPR*. 2017.

[17] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. «Unsupervised Visual Representation Learning by Context Prediction». In: *ICCV*. 2015.

[18] Mehdi Noroozi and Paolo Favaro. «Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles». In: *ECCV*. 2016.

[19] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. «Unsupervised Representation Learning by Predicting Image Rotations». In: *ICLR*. 2018.

[20] Simon Jenni and Paolo Favaro. «Self-supervised feature learning by learning to spot artifacts». In: *CVPR*. 2018, pp. 2733–2742.

[21] Mathilde Caron et al. «Deep Clustering for Unsupervised Learning of Visual Features». In: *ECCV*. 2018.

[22] Zhongzheng Ren and Yong Jae Lee. «Cross-Domain Self-supervised Multi-task Feature Learning using Synthetic Imagery». In: *CVPR*. 2018.

[23] Andrew Owens et al. «Learning Sight From Sound: Ambient Sound Provides Supervision for Visual Learning». In: *IJCV*. 2018.

[24] Ishan Misra, C. Lawrence Zitnick, and Martial Hebert. «Shuffle and Learn: Unsupervised Learning using Temporal Order Verification». In: *ECCV*. 2016.

[25] Lerrel Pinto et al. «The Curious Robot: Learning Visual Representations via Physical Interactions». In: *ECCV*. 2016, pp. 3–18.

[26] Jiaxin Li, Ben M. Chen, and Gim Hee Lee. «SO-Net: Self-Organizing Network for Point Cloud Analysis». In: *CVPR*. 2018.

[27] Yongheng Zhao et al. «3D Point Capsule Networks». In: *CVPR*. 2019.

[28] Yaoqing Yang et al. «FoldingNet: Point Cloud Auto-Encoder via Deep Grid Deformation». In: *CVPR*. 2018.

[29] Zhizhong Han et al. «Multi-Angle Point Cloud-VAE: Unsupervised Feature Learning for 3D Point Clouds From Multiple Angles by Joint Self-Reconstruction and Half-to-Half Prediction». In: *ICCV*. 2019.

[30] Ling Zhang and Zhigang Zhu. «Unsupervised Feature Learning for Point Cloud by Contrasting and Clustering With Graph Convolutional Neural Network». In: *CVPR Workshop*. 2019.

[31] Jonathan Sauder and Bjarne Sievers. «Self-Supervised Deep Learning on Point Clouds by Reconstructing Space». In: 2019.

[32] Ali K. Thabet, Humam Alwassel, and Bernard Ghanem. «MortonNet: Self-Supervised Learning of Local Features in 3D Point Clouds». In: *Prepring ArXiv abs-1904-00230* (2019).

[33] Kaveh Hassani and Mike Haley. «Unsupervised Multi-Task Feature Learning on Point Clouds». In: *ICCV*. 2019.

[34] Mingsheng Long et al. «Learning Transferable Features with Deep Adaptation Networks». In: *International Conference on Machine Learning (ICML)*. 2015.

[35] Mingsheng Long et al. «Deep Transfer Learning with Joint Adaptation Networks». In: *ICML*. 2017.

[36] Baochen Sun and Kate Saenko. «Deep CORAL: Correlation Alignment for Deep Domain Adaptation». In: *ECCV Workshop*. 2016.

[37] Shai Ben-David et al. «A theory of learning from different domains». In: *Machine Learning* 79 (2010), pp. 151–175.

[38] Massimilano Mancini et al. «Boosting Domain Adaptation by Discovering Latent Domains». In: *CVPR*. 2018.

[39] Fabio Maria Carlucci et al. «AutoDIAL: Automatic DomaIn Alignment Layers». In: *ICCV*. 2017.

[40] Yaroslav Ganin et al. «Domain-adversarial Training of Neural Networks». In: *J. Mach. Learn. Res.* 17.1 (2016), pp. 2096–2030.

[41] Eric Tzeng et al. «Adversarial Discriminative Domain Adaptation». In: (2017).

[42] Paolo Russo et al. «From source to target and back: symmetric bi-directional adaptive GAN». In: *CVPR*. 2018.

[43] Judy Hoffman et al. «CyCADA: Cycle-Consistent Adversarial Domain Adaptation». In: *ICML*. 2018.

[44] Muhammad Ghifary et al. «Domain Generalization for Object Recognition with Multi-task Autoencoders». In: *ICCV*. 2015.

[45] Haoliang Li et al. «Domain Generalization With Adversarial Feature Learning». In: *CVPR*. 2018.

[46] Konstantinos Bousmalis et al. «Domain Separation Networks». In: *NIPS*. 2016.

[47] Saeid Motiian et al. «Unified Deep Supervised Domain Adaptation and Generalization». In: *ICCV*. 2017.

[48] Ruijia Xu et al. «Larger Norm More Transferable: An Adaptive Feature Norm Approach for Unsupervised Domain Adaptation». In: *ICCV*. 2019.

[49] Ya Li et al. «Deep Domain Generalization via Conditional Invariant Adversarial Networks». In: *ECCV*. 2018.

[50] Fabio Maria Carlucci et al. «Hallucinating Agnostic Images to Generalize Across Domains». In: *ICCV Workshop*. 2019.

[51] Shiv Shankar et al. «Generalizing Across Domains via Cross-Gradient Training». In: *ICLR*. 2018.

[52] Riccardo Volpi et al. «Generalizing to Unseen Domains via Adversarial Data Augmentation». In: *NIPS*. 2018.

[53] Da Li et al. «Learning to Generalize: Meta-Learning for Domain Generalization». In: *AAAI*. 2018.

[54] Da Li et al. «Episodic Training for Domain Generalization». In: *ICCV*. 2019.

[55] Fabio M. Carlucci et al. «Domain Generalization by Solving Jigsaw Puzzles». In: *CVPR*. 2019.

[56] Jiaolong Xu, Liang Xiao, and Antonio M. López. «Self-supervised Domain Adaptation for Computer Vision Tasks». In: *Preprint Arxiv* abs/1907.10915 (2019).

[57] Xiaohua Zhai et al. «S4L: Self-Supervised Semi-Supervised Learning». In: *ICCV*. 2019.

[58] Tuan-Hung Vu et al. «ADVENT: Adversarial Entropy Minimization for Domain Adaptation in Semantic Segmentation». In: *CVPR*. 2019.

[59] Liang Du et al. «SSF-DAN: Separated Semantic Feature Based Domain Adaptation Network for Semantic Segmentation». In: *ICCV*. 2019.

[60] Stephen James et al. «Sim-To-Real via Sim-To-Sim: Data-Efficient Robotic Grasping via Randomized-To-Canonical Adaptation Networks». In: *CVPR*. 2019.

[61] Can Qin et al. «PointDAN: A Multi-Scale 3D Domain Adaption Network for Point Cloud Representation». In: *Advances in Neural Information Processing Systems 32*. 2019.

[62] Mikaela Angelina Uy et al. «Revisiting Point Cloud Classification: A New Benchmark Dataset and Classification Model on Real-World Data». In: *Proc. ICCV*. 2019.

[63] Longlong Jing and Yingli Tian. «Self-supervised Visual Feature Learning with Deep Neural Networks: A Survey». In: *arXiv:1902.06162 [cs]* (Feb. 16, 2019). arXiv: 1902.06162. URL: http://arxiv.org/abs/1902.06162 (visited on 11/03/2019).

[64] Richard Zhang, Phillip Isola, and Alexei A. Efros. «Split-Brain Autoencoders: Unsupervised Learning by Cross-Channel Prediction». In: *CVPR*. 2017.

[65] Xinrui Zhuang et al. «Self-supervised Feature Learning for 3D Medical Images by Playing a Rubik's Cube». In: *MICCAI*. 2019.

[66] Rich Caruana. «Multitask Learning». In: *Mach. Learn.* 28.1 (1997), pp. 41–75. ISSN: 0885-6125. DOI: 10.1023/A:1007379606734. URL: https://doi.org/10.1023/A:1007379606734.

[67] Long Duong et al. «Low Resource Dependency Parsing: Cross-lingual Parameter Sharing in a Neural Network Parser». In: *IJCNLP*. 2015.

[68] Ishan Misra et al. *Cross-stitch Networks for Multi-task Learning*. 2016. eprint: arXiv:1604.03539.

[69] Yongxin Yang and Timothy M. Hospedales. *Trace Norm Regularised Deep Multi-Task Learning*. 2016. eprint: arXiv:1606.04038.

[70] Sebastian Ruder. *An Overview of Multi-Task Learning in Deep Neural Networks*. 2017. eprint: arXiv:1706.05098.

[71] Sinno Jialin Pan and Qiang Yang. «A Survey on Transfer Learning». In: *IEEE Trans. on Knowl. and Data Eng.* 22.10 (2010), pp. 1345–1359. ISSN: 1041-4347. DOI: 10.1109/TKDE.2009.191.

[72] Yaroslav Ganin and Victor Lempitsky. «Unsupervised Domain Adaptation by Backpropagation». In: (2014).

[73] Da Li et al. *Deeper, Broader and Artier Domain Generalization*. 2017. eprint: arXiv:1710.03077.

[74] Haoliang Li et al. «Domain Generalization With Adversarial Feature Learning». In: *CVPR*. 2018.

[75] Aditya Khosla et al. «Undoing the Damage of Dataset Bias». In: *ECCV*. Florence, Italy, 2012.

[76] Eman Ahmed et al. «Deep Learning Advances on Different 3D Data Representations: A Survey». In: *CoRR* abs/1808.01462 (2018). URL: http://arxiv.org/abs/1808.01462.

[77] Chiyu Max Jiang et al. «Convolutional Neural Networks on non-uniform geometrical signals using Euclidean spectral transformation». In: *CoRR* (2019). arXiv: 1901.02070. URL: http://arxiv.org/abs/1901.02070.

[78] Zhiyuan Zhang et al. *Rotation Invariant Convolutions for 3D Point Clouds Deep Learning*. 2019. eprint: arXiv:1908.06297.

[79] Li Yi et al. «A scalable active framework for region annotation in 3D shape collections». In: *ACM Transactions on Graphics* 35 (2016), pp. 1–12. DOI: 10.1145/2980179.2980238.

[80] Travis E Oliphant. *A guide to NumPy*. Vol. 1. Trelgol Publishing USA, 2006.

[81] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. «Open3D: A Modern Library for 3D Data Processing». In: *arXiv:1801.09847* (2018).

[82] Binh-Son Hua et al. «SceneNN: A Scene Meshes Dataset with aNNotations». In: *3DV*. 2016.

[83] Angela Dai et al. «ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes». In: *CoRR* (2017). URL: http://arxiv.org/abs/1702.04405.