POLITECNICO DI TORINO

Master degree in Computer engineering

Optimal grasp point generation for robotic object manipulation using PointNet++

Supervisors

Prof.essa Marina Indri

Supervisors at Comau

Ing. Simone Panicucci

Ing. Enrico Civitelli

Candidate

Giuseppe Masiello

December 2023

Abstract

Robotic object manipulation has seen remarkable advancements in recent years, with the potential to revolutionize industries ranging from manufacturing to healthcare. A critical aspect of successful manipulation is the generation of optimal grasp points, which directly influence the efficiency and reliability of robotic operations. This thesis delves into the domain of grasp planning and introduces a novel approach to grasp point generation leveraging the PointNet++ architecture.

The proposed methodology harnesses the power of deep learning and point cloud data processing to identify and select grasp points that maximize the probability of success. Through an extensive review of the existing literature, we identify the limitations of traditional grasp planning techniques and highlight the advantages of utilizing deep learning models and point cloud data processing for this purpose.

In this study, we present the implementation and evaluation of our PointNet++-based grasp point generation system. We describe the data collection and preprocessing procedures, the network architecture, and the training process, demonstrating the efficacy of our approach in real-world scenarios. We assess the system's performance with respect to grasp success rate, execution speed, and generalizability to novel objects.

This work contributes to the ongoing efforts to make robots more versatile and capable in handling complex and unstructured environments, with implications for industries such as logistics, warehousing, and healthcare, among others.

Table of Contents

List of Figures

Chapter 1

Introduction

Robotics is an ever-evolving field that plays an increasingly significant role in a wide range of applications, from industrial manufacturing to robot-assisted surgery. At the core of many of these contexts, the ability to manipulate objects is a critical skill that can enhance operational efficiency, reduce human risk, and open up new possibilities in previously inaccessible domains. However, robotic object manipulation represents a complex challenge, requiring the integration of expertise in engineering, computer science, and artificial intelligence.

The fundamental starting point for success in object manipulation is a robot's capacity to identify and select optimal grasp points on an object. A well-chosen grasp point can ensure stability, safety, and control during the manipulation task, enabling the robot to perform the task with precision and reliability. The generation of such optimal grasp points is a pivotal aspect of grasp planning and forms the core focus of this thesis.

In this research, we explore two distinct approaches for grasp point generation. The first approach involves classification, where we aim to classify potential grasp points as either suitable or unsuitable for manipulation. This classification approach offers a binary decision on the grasp quality of points. The second approach takes a regression perspective, seeking to predict the coordinates of the optimal grasp point directly.

To comprehensively evaluate the effectiveness of these approaches, we employ two different samplers. The first utilizes an adaptive matrix selection method generated by a small neural network that selects the most suitable points for grasp. This neural network-based matrix selection process enables adaptation to the object's shape and characteristics, offering a dynamic solution for optimal grasp point generation.

The second sampler leverages the Chamfer distance metric, striving to minimize the distance between the sampled points and the points within the point cloud representing the object's surface. This approach ensures that the selected grasp points closely align with the object's geometry, enhancing the precision of grasp point identification. The Chamfer distance-based sampler optimizes the selection process, taking into account the object's surface and geometry for effective grasp point generation.

These dual sampling strategies enable us to investigate not only the choice of grasp point generation model but also the method of sampling the candidate points for evaluation. The selection of these two diverse approaches and samplers represents a crucial aspect of this research, promising a comprehensive analysis of the grasp point generation process.

This thesis aims to provide a deep understanding of the optimal grasp point generation process for robotic object manipulation.

The subsequent chapters of this thesis will delve into the methodologies, experiments, and results that underpin our analysis of grasp point generation using PointNet++.

1.1 State of art

Before delving into our research methodologies and findings, it is essential to provide a comprehensive overview of the current state of the art in the field of robotic object manipulation and grasp point generation. In this paragraph, we will navigate through the existing literature, highlighting the key advancements, challenges, and trends that have shaped the landscape of grasp point generation.

Dex-Net 3.0[1] is a valuable dataset of point clouds, particularly significant in the context of vacuum gripper-based robotic manipulation. It distinguishes itself as one of the few datasets tailored to assess grasp points in the context of vacuum grippers. However, it is important to highlight a notable limitation of this dataset: each point cloud has only one grasp point classified. This constraint can render Dex-Net 3.0 less efficient when compared to datasets that offer multiple classified grasp points per point cloud. We will analyze the details of this dataset in the next chapter. To harness the full potential of Dex-Net 3.0, its creators have developed a neural network known as **GQCNN**, short for Grasping Quality Convolutional Neural Network. This neural network takes the depth image and the position of the grasp point as input and is designed to classify and evaluate that point. In this approach the grasp point is always in the center of the depth image, so we need to compute only the orientation through the point cloud.

SuctionNet 1-billion[2] is also a dataset of point clouds, unlike DexNet contains dense annotation for grasping points. The authors conducted a series of experiments employing a convolutional neural network known as '**DeepLabV3**' [3] this network takes as input the depth image of the scene and through 2D convolution computes two maps: the first is the probability of suction cup closing,

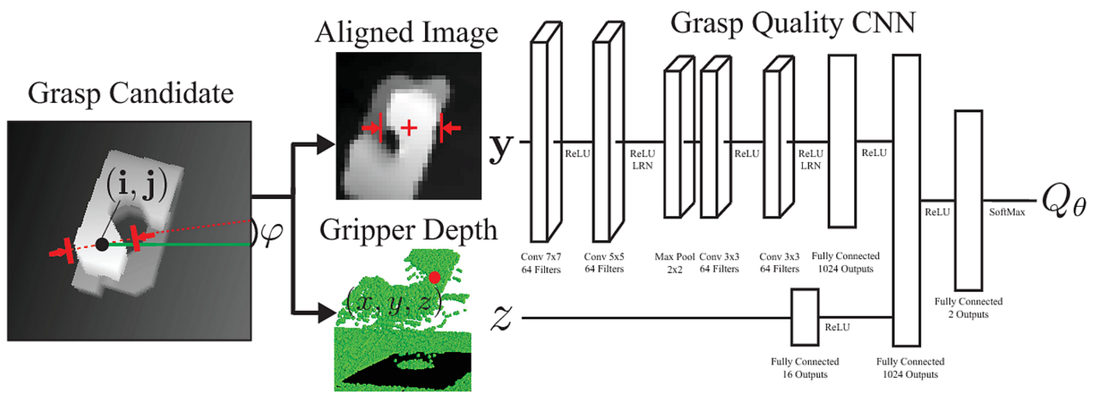

Figure 1.1: GQCNN network proposed to evaluate DexNet3.0 dataset. Source [1]

the latter is the center of the object, each of these two maps contains values between 0 and 1. The two maps are then multiplied to create a map that predicts the best point for grasping an object. To compute the orientation of the grasp this approach exploits the point cloud to estimate the normals.

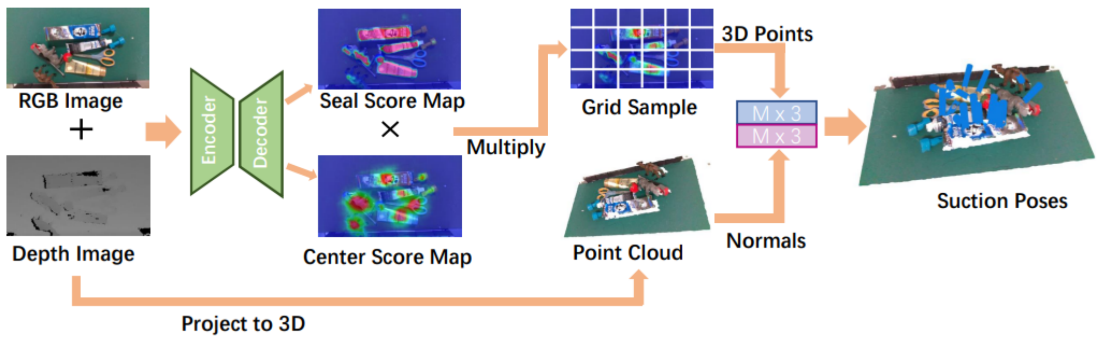

Figure 1.2: Proposed pipeline. Encoder and decoder are part of the cited DeepLabV3. It can be seen how they use point cloud to estimate normals and the two maps returned from the network. Source [2]

Big-Net[4] like GQCNN and DeepLabV3 utilizes the 2D convolution layer to learn the best point suitable for grasping. This network is an encoder-decoder that predicts a set of parameter for grasping: the orientation, the quality of each point and a gripper step. This last parameter is interesting, because the network tries to learn in which area the surface of the object is the smoothest, however this parameter changes accordingly to the gripper, so the network should be re-trained with a different grippers. The base block of Big-Net is the ResNet block except for the branch that works on the depth image that exploits CBAMs to improve the robustness of the network.

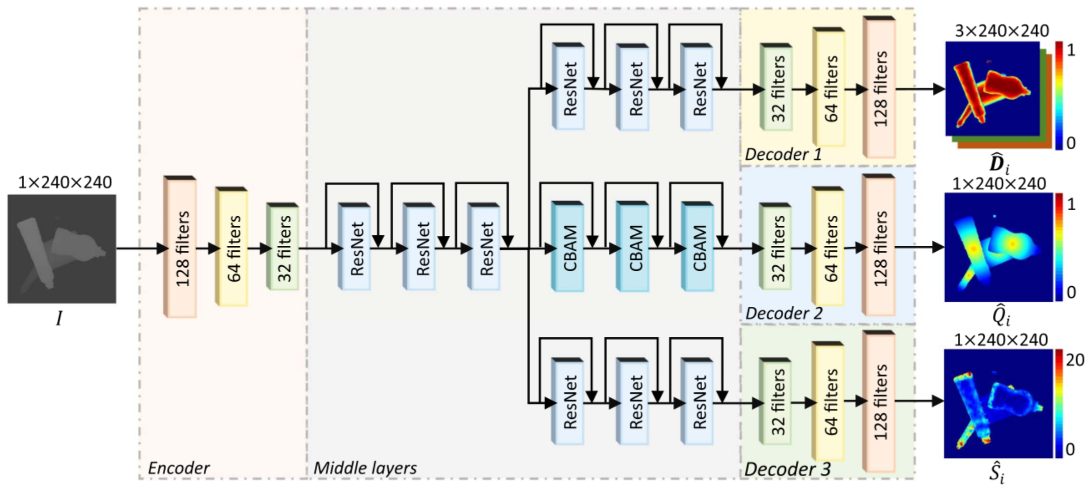

Figure 1.3: Architecture of Big-Net, the encoder and decoder has the same parameters in the inverse order respect the other. In the middle layers there are ResNet and CBAMs blocks. Source [4]

We can infer that these approaches use only part of the information from the point cloud, as they employ methods commonly used in image processing to accomplish the task. They use the point cloud solely to estimate the orientation for grasping. The approach we want to pursue relies exclusively on the point cloud and, therefore, employs innovative processing techniques.

A similar approach to what we want to pursue is **Learning2grasp** [5]. In that work the task to be carried out is to grasp the objects with a gripper with parallel jaws unlike ours, which works with a vacuum gripper. To do so, they employ a neural network made up by PointNet++ as encoder, followed by a sampler to reduce the number of points of input and, due to the nature of the gripper equipped with two jaws, there is also a regressor to predict the second point of contact. As final step, all the point extracted by the sampler are then classified. In this way the network returns as output M points with the 6 coordinates of the two contact points and the orientation of the grasp, each of them with a score that indicates the probability of success of the grasp.

1.2 Structure of thesis

The thesis is structured as follows:

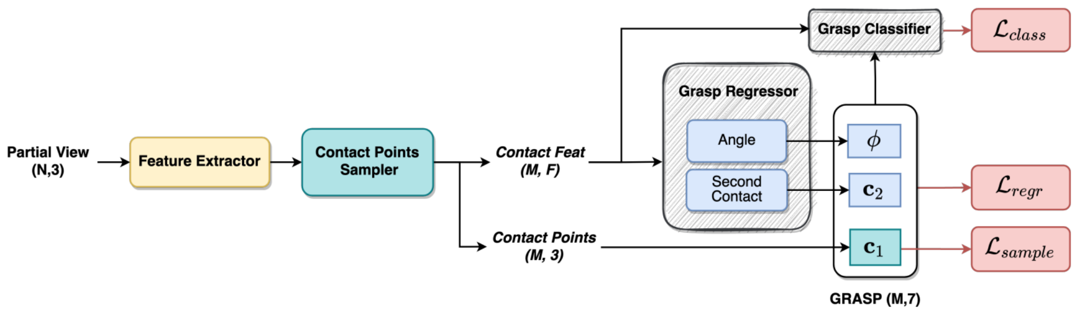

Figure 1.4: Scheme of the proposed network in Learning to Grasp. Feature extractor is the PointNet++ backbone and contact sampler is a sampler based on Chamfer distance. Grasp classifier and Grasp regressor are two small network that performs classification and regression of the second contact point, respectively. Source [5]

- Chapter 2 provides an overview of the mathematical foundations of grasping, an introduction to the coordinates system and how we apply transformation to pass from one to another.
- Chapter 3 describes the encoder used in our network, with a focus on explaining what a ‘point cloud’ is within the context of this work.
- Chapter 4 explores two techniques used for sampling point clouds, with detailed explanations.
- Chapter 5 describes the two dataset used to train the neural network
- Chapter 6 describes the two specific approaches used in the thesis to address the task.
- Chapter 7 compares the results obtained from the application of the two approaches, with analysis and evaluations.
- Chapter 8 summarizes the main conclusions of the work and discuss potential future directions for further research.

Chapter 2

Formalization of a Grasp Point in 3D Space with Vacuum Gripper

In the context of robotics and automation, the grasping process plays a fundamental role in handling objects in the physical world. This chapter focuses on the formalization of what a grasp point is in three-dimensional space, with particular attention to a specific type of gripper: the vacuum gripper. Then we focus on the transformation from a grasp point in image coordinates system to a grasp point in robot coordinates system relative to its base.

2.1 Introduction to 3D Grasping

Grasping, or gripping, is the act by which a robot or manipulative machine acquires an object and takes control of it. This process requires the determination of a grasp point, which is the point in 3D space where the gripper can establish an effective grip. The formalization of a grasp point involves various aspects, including the object's geometry, gripper characteristics, and the surrounding environment.

2.2 Vacuum Gripper

The vacuum gripper is a gripping tool that uses suction to grasp objects. It consists of a suction cup that can create a pressure difference between the interior and exterior, allowing it to adhere to the object to be grasped. This type of gripper is widely used for flat objects such as sheets, plates, or objects with flat and smooth surfaces.

2.3 Formalization of a Grasp Point

To formalize a grasp point in 3D space with a vacuum gripper, we need to consider several factors. These include:

- **Object Geometry**: The object's geometry is fundamental in determining the grasp point. We should consider the shape, dimensions, and surface of the object. In the case of a vacuum gripper, it is essential that the object has a sufficiently flat and smooth surface to which the suction cup can adhere effectively.

- **Gripper Position and Orientation**: The gripper's position and orientation relative to the object must be precisely defined. These parameters will influence where the suction cup can be applied successfully. Additionally, we should consider the robot's or manipulative machine's kinematics to determine these positions and orientations.

- **Material and Surface Properties**: The material and surface properties of the object will affect the vacuum gripper's ability to create an effective seal. Porosity, roughness, and compatibility with suction are critical factors to consider.

These factors are essential for achieving precise and efficient grasping operations in the world of robotics and automation.

2.4 Coordinate systems

Let's start by defining what **image coordinates** are. These coordinates are typically utilized to represent the position of an object in the image plane captured by a camera or visual sensor. Image coordinates are usually expressed in pixels and may vary based on the image resolution and the origin, typically located at the top-left corner.

Camera coordinates, on the other hand, constitute a three-dimensional coordinate system that defines the camera's position and orientation relative to the observed object or environment. This system enables the precise determination of object positions within the images captured by the camera, facilitating the calculation of their three-dimensional real-world positions.

Robot coordinates, in relation to the robot's base, represent the robot's position and orientation within a physical reference system. This reference system is firmly anchored to the robot's base, enabling accurate determination of the robot's gripper's position and orientation.

2.5 Transformation from Image to Robot Coordinates

Transforming a grasp point from image coordinates to robot coordinates relative to the base requires the accurate calibration between the camera (or visual sensor) and the robot. Key steps for this transformation include:

- **Calibration**: Accurate calibration is required to establish the relationship between the image coordinate system and the robot's coordinate system relative to the base. The calibration process is typically divided into two steps: **intrinsic calibration** and **extrinsic calibration**. Intrinsic calibration involves the determination of the camera's internal parameters, such as radial and tangential distortion. This is often achieved using well-known calibration patterns, such as calibration grids. The matrix used is in the form:

  $$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

  where f_x and f_y are the focal length of the camera, c_x and c_y represent the camera principal points and s is the skew.

  The intrinsic calibration matrix is essential as it allows for an accurate mapping between the coordinates in image space captured by the camera and the actual 3D spatial coordinates, thus enabling the precise determination of the three-dimensional positions of points based on their two-dimensional representation in the image.

  The extrinsic calibration, instead, involves measuring known reference points or using visual markers, and through a system of linear equations computes the transformation matrix. The extrinsic calibration matrix, on the other hand, performs the mapping between the coordinates in the camera's coordinate system and the coordinates in the robot's coordinate system, facilitating the seamless integration of visual data with the robot's spatial framework.

  $$T_{calibration} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tag{2.2}$$

  where R is a matrix 3×3 which indicates the rotation between the two coordinate systems and t is a translation vector 3×1 that indicates the distance between the origins of the two coordinate systems. So the resulting matrix $T_{calibration}$ is 4×4.

- **Coordinate Transformation**: Once the camera is calibrated to the robot, a transformation matrix can be applied to convert image coordinates into robot coordinates. In order to compute the coordinates of a point with respect to the robot's reference system we need to compute the product between the calibration matrix and the point with respect to the image coordinate system.

$$p_{robot} = T_{calibration} * p_{image} \tag{2.3}$$

After the transformation, it is possible to calculate the grasp point in robot coordinates relative to the base. This point represents the position and orientation that the gripper must reach to successfully grasp the object.

Chapter 3

PointNet and PointNet++ in 3D data processing

In the realm of artificial intelligence (AI), point clouds represent a vital data structure for the analysis of three-dimensional environments.

A point cloud is a mathematical representation of unstructured and discrete three-dimensional (3D) data. Formally, a point cloud is defined as a set of points in three-dimensional space (3D), where each point is described by a three-dimensional vector of coordinates. This set can be represented as follows:

$$P = \{(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots, (x_n, y_n, z_n)\} \tag{3.1}$$

P represents the point cloud under consideration, n is the total number of points contained in the point cloud and the x, y, and z coordinates of each point represent the position of the point in the three-dimensional environment. The point cloud can consist of a variety of points with different positions and densities, thus representing the surface or structure of physical objects, real environments, or abstract three-dimensional entities. Point clouds have found widespread application in AI, particularly in computer vision and machine learning. Operations on point clouds involve a range of tasks, including point cloud registration, segmentation, feature extraction, and object recognition. Point cloud data, often acquired through LiDAR sensors or 3D cameras, are invaluable for applications like autonomous navigation, robotics, environmental modeling, and augmented reality. Despite their raw and unstructured nature, point clouds are processed using specialized AI techniques, such as deep learning architectures like PointNet and PointNet++, to extract meaningful patterns and features.

3.1 PointNet

PointNet is the first framework for 3D data processing that is proposed by Charles R. Qi [6]. This innovative approach faces different challenges in processing of non structured data, such as point clouds generated from LiDAR sensors or 3D cameras. The main feature of PointNet is the capability to process 3D data without the need to apply complex transformation; to achieve this it exploits two properties:

- permutation invariance: due to the fact that point clouds are unordered set of point.
- transformation invariance: rotating and translating points should not affect the behaviour of the network.

In practice, to achieve permutation invariance PointNet takes advantage of pooling operation, a symmetric function used to aggregate the features from each point.
The transformation invariance, instead, is obtained through two mini-networks (T-net) that predict an affine transformation that tries to align different input shapes and also different feature spaces.
In PointNet, as illustrated in Figure 3.1, the initial step involves achieving transformation invariance for both the input data and features. Following this, the network employs multi-layer perceptrons to extract relevant features. These extracted features are then aggregated using a max-pooling operation. The final output consists of classification scores. PointNet for segmentation, on the other hand, diverges somewhat from this process. Instead of classifying global features, the network concatenates the global feature vector with the features obtained from the feature transformation step. Subsequently, it applies multi-layer perceptrons to process this concatenated information. The output is a feature vector for each individual point within the input data.

3.2 PointNet++

PointNet++[7] is an extention of PointNet; it was developed to increase the ability to extract feature and relationships between points in the point cloud. One of the key feature of PointNet++ is the ability to learn hierarchical representation of point cloud, which allows the network to acquire information on different spatial scales. This is done through:

- Sampling: in this phase a subset of point cloud is selected and these points defined the centroid. The sampling is implemented in a smart way to capture relevant points, in order to keep main feature of the point cloud. One of the method to make sampling is furthest point sampling, which is an iterative

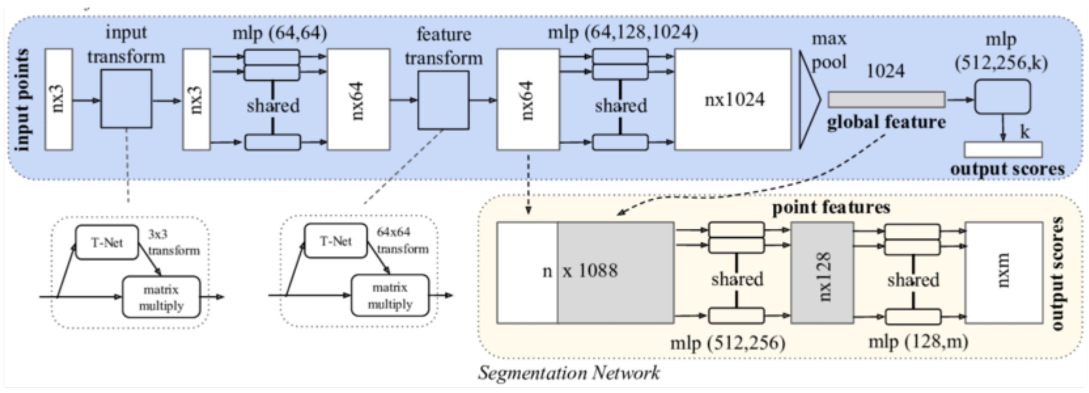

Figure 3.1: PointNet - architecture of PointNet. The blue part stand for the classification task, while the yellow part is for segmentation. Source [6]

method, which at each step extracts the furthest point from the subset of points already selected. In this way, there is a uniform coverage of the initial point cloud.

- Grouping: in this phase the network finds local regions around each centroid through a ball query. Each ball query selects at least K points within radius R

- PointNet: in this phase a PointNet is applied to each group to extract local feature.

This three operations are applied hierarchically, diminishing the number of points extracted by sampling operation, and increasing the number of K points nearest each centroids. In this way, the network can learn pattern at different scale, focusing on a different part at each iteration.
It is common for data structures, such as point clouds, to have points distributed unevenly across the area. This particularity introduces a challenge for the network, because it can generalize well on regions with high density points but can face difficulties in sparse areas. To deal with this issues, Charles R.Qi introduces ‘Multi scale grouping’ [7] in which the ball query operation is applied to each grouping layer with different values of K and consequently of the radius. In this way, each hierarchical level looks at different local regions.
In sampling layer the number of point is decreased; however, in segmentation task we want to obtain per-point feature for all the original points. To cope with this problem, feature propagation operation is introduced. This operation consists of propagate features by interpolating features from the last layer of PointNet into the previous one, so from bottom to top. Interpolation is done by using inverse

distance weighted average based on k nearest neighbors as follows:

$$f^{(j)}(x) = \frac{\sum_{i=1}^{k} w_i(x) f_i^{(j)}}{\sum_{i=1}^{k} w_i(x)} \quad where \quad w_i(x) = \frac{1}{d(x, x_i)^p}, \quad j = 1, \ldots, C \qquad (3.2)$$

The interpolated features are then concatenated with features of that level and passed through PointNet. In this way, PointNet++ attains also per-point feature.

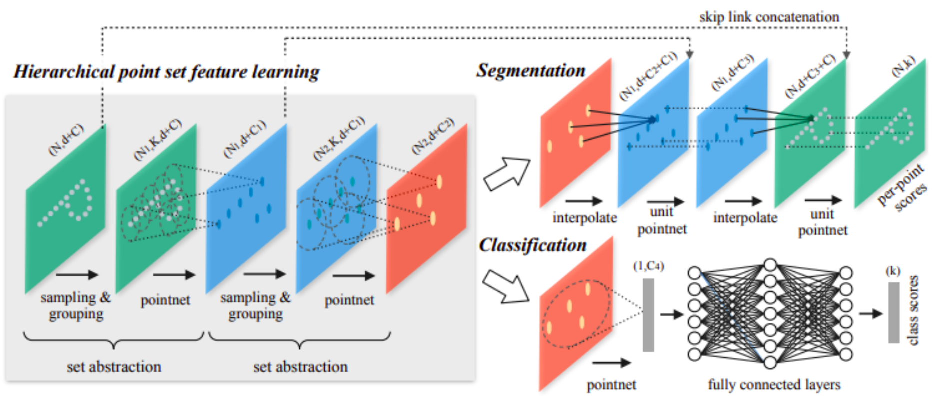

Figure 3.2: PointNet++ - Overall architecture of PointNet++ the upper stream is for segmentation of point cloud, the lower stream is for classification. Source [7]

Chapter 4

Sampling point in 3D data

In the realm of point cloud processing, the sampling process plays a critical role in selecting relevant points for specific analysis and processing tasks. Sampling a point cloud involves selecting a subset of points from a point cloud P, where P is defined as in 3.1.

Sampling aims at creating a subset S with k points, where $k \leq n$, to satisfy a specific objective or criterion. The subset is defined as

$$S = \{(x_{i_1}, y_{i_1}, z_{i_1}), (x_{i_2}, y_{i_2}, z_{i_2}), \ldots, (x_{i_k}, y_{i_k}, z_{i_k})\} \tag{4.1}$$

where $i_1, i_2, \ldots, i_k$ represent the indices of the points selected from the point cloud P.

In this chapter, we will explore two distinct approaches to point cloud sampling, each adopting a unique strategy for point selection.

The first approach is based on the 'Chamfer distance', a metric that measures the distance between two sets of points. This sampling method aims at selecting points that minimize the discrepancy between the original point cloud and a sampled version, providing an approximate representation of the 3D object or environment under consideration. Through an analysis of the advantages and limitations of this approach, we will examine how the Chamfer distance can be used to extract meaningful information from point clouds.

The second approach we will explore is based on the generation of a selection matrix. In this case, an algorithm creates a matrix that represents which points should be sampled and which should be discarded. This selection matrix is designed to maximize computational efficiency and the effectiveness of subsequent point cloud processing operations. We will see how this approach can contribute to simplify point cloud sampling and analysis, streamlining the point selection process.

Through a detailed comparative analysis of these two sampling methods, we will highlight situations where each one excels and those where they may present some limitations.

4.1 Chamfer distance

The Chamfer Distance is a distance metric used to measure the dissimilarity between two point sets, such as the sampled subset (S) and the original point cloud (P). It quantifies how well the sampled subset approximates the original point cloud and is defined by the following equation:

$$\mathcal{L}_{Chamfer}(S,P) = \frac{1}{|P|}\sum_{p\in P}\min_{s\in S}\|p-s\|_2^2 + \frac{1}{|S|}\sum_{s\in S}\min_{p\in P}\|s-p\|_2^2 \tag{4.2}$$

Where S represents the sampled subset of points, P is the original point cloud and $\|\cdot\|_2$ denotes the Euclidean norm, which calculates the straight-line distance between two points in 3D space.

The two summations account for the distances between points in both directions: from points in P to their nearest neighbors in S, and vice versa. This bilateral approach makes Chamfer Distance more robust and capable of detecting symmetric discrepancies between the two sets. It means that the metric takes into account differences both from points in P to points in S and from points in S to points in P, ensuring that discrepancies in both directions are equally considered. It also helps to avoid an unbalanced solution, where one set approximates the other well in one direction but not in the other. The goal is to create a subset S that is as balanced as possible compared to P, allowing for the most comprehensive representation of the original set.

To speed up the convergence of (4.2), an additional term is added; this term correspond to the maximum distance between the set of sampled point and the point cloud, as follows:

$$\mathcal{L}_{simplify}(S,P) = \mathcal{L}_{chamfer} + \max_{s\in S}\min_{p\in P}\|s-p\|_2^2 \tag{4.3}$$

In this way also the worst case is minimized. We call this term 'simplification loss'.

4.1.1 Projection

When performing point cloud sampling, it is important to note that the set of generated points may not exactly form a subset of the original point cloud This is due to the fact that the two subsets are only close to each other in euclidean space. As a result, a matching operation [8] is required to establish correspondence between the extracted points and those from the source point cloud. This matching process is typically accomplished using a Nearest Neighbor search. Nearest Neighbor search involves finding the point in the original cloud that is the closest to each generated point and replacing it. Since multiple points in the generated set G might be the closest to the same point in the original point cloud P, it is possible that the number of unique sampled points could be smaller than the requested sample size. This matching operation unfortunately is not differentiable and cannot backpropagate the loss, so it is applied only during inference time.

4.1.2 Soft-projection

To address the problem of non-differentiability, Itai Lang et Al. [9], proposed ‘soft-projection’ operation. In this operation, each point s in a set S of sampled points is softly projected using a weighted average of the k nearest neighbors as follows:

$$r = \sum_{i \in N_P(S)} w_i p_i \tag{4.4}$$

where $\mathrm{N_p(S)}$ is the set that contains the index of the k points near the sampled point. We can see r as the expected value of a probability distribution formed by points p_i with weights w_i.

The weights are computed based on the euclidean distance between s and its neighbors.

$$w_i = \frac{e^{-d_i^2/t^2}}{\sum_{j \in N_p(q)} e^{-d_j^2/t^2}} \quad where \quad d_i = \|s - p_i\|_2 \tag{4.5}$$

In equation (4.5) there is also a learnable temperature parameter t; this term controls the shape of distribution. In fact, when $t \to 0$ the distribution converges to a Kronecker delta function and r converges to its nearest neighbors. To achieve this, it is added a projection loss .

$$\mathcal{L}_{projection} = t^2 \tag{4.6}$$

In this way the loss promotes a small value of t.

In Figure 4.1 can be seen a scheme of the overall process.

4.1.3 Network

This section illustrate the structure of the network used for point sampling. In Figure (4.2) there is the scheme proposed by Itai Lang [9].

The simplification network performs the sampling of the points through a neural network made up of a ‘multi-layers perception’, a pooling operation and a fully connected layer. This network returns the set of sampled point $\mathrm{S_{m\ x\ 3}}$, to which

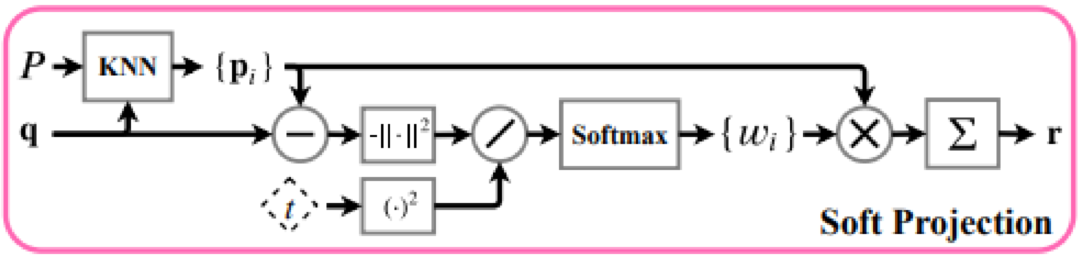

Figure 4.1: Scheme of the Soft-Projection process. Source [9].

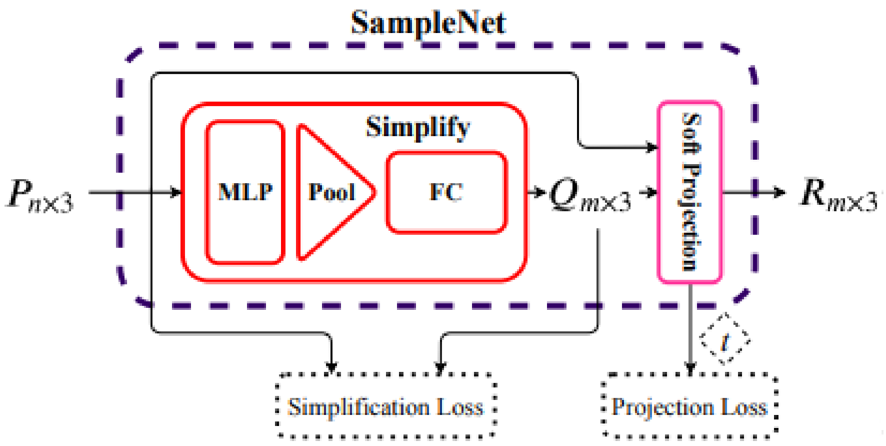

Figure 4.2: SampleNet - Scheme of the overall network used to sample point. Source [9].

the simplification loss is applied (4.3). This set is then soft-projected accordingly to what described before and the projection loss is applied (4.6). The final loss of the sampleNet is:

$$\mathcal{L}_{total} = \mathcal{L}_{simplification} + \mathcal{L}_{projection} \tag{4.7}$$

4.2 Selection matrix

In the field of point cloud representation and analysis, the use of selection matrices represents an innovative approach to perform sampling. Point clouds, being three-dimensional sets of points, often contain a vast amount of data, and for processing and analysis purposes, it may be necessary to extract a significant and representative subset of these points.

Selection matrices are employed for this purpose by assigning weights or probabilities to each point in the point cloud. In other words, each point is treated as an element in a weight vector. These weights determine the likelihood of including the point in the sampled version of the point cloud. Mathematically, the sampling operation can be described as a multiplication between the selection matrix S and the original point cloud P, resulting in the sampled subset P_s:

$$P_s = S * P \tag{4.8}$$

This matrix has m × n dimensions, where m (m < n) is the number of sampled points and n is the number of points of the original point cloud. Each row of the selection matrix is a one hot vector that indicates which point of the original point cloud is sampled. Each column represents how many times that point is sampled and in which point $p_s \in P_s$ is mapped.

Called $p_{i,j}$ a generic element of the matrix S, where i indicates the ith row and j indicates the jth column, $p_{i,j}$ is given by:

$$p_{i,j} = \begin{cases} 1 & \text{if j = index of selected point} \\ 0 & \text{otherwise} \end{cases} \tag{4.9}$$

The matrix should select only one point for each row so:

$$\sum_{j=1}^{N} (p_{i,j}) = 1 \tag{4.10}$$

Equation (4.9) and (4.10) allow the matrix to be a selection one.

4.2.1 Method

The network architecture proposed by Wang M. [10] contains only few layers to keep the complexity of the network low. This network creates the matrix from the feature extracted by a backbone instead of extract it from the point cloud like the other method. As it can be seen from Figure (4.3) the network is composed of 'multi-layer perception' (MLP) and a pooling layer to extract global feature (F_{global}); this two outputs are then concatenated to construct a vector with per-point feature (F_{local}) and global feature:

$$F_{global} = MaxPool(F_{local}) \tag{4.11}$$

Then, there are two MLP and lastly the sigmoid operation is applied:

$$\hat{S} = f(f(concatenate(F_{local}, F_{global}))) \tag{4.12}$$

$$\hat{S} = sigmoid(\hat{S}) \tag{4.13}$$

The output of the sigmoid is a matrix with the same dimensions of the selection matrix S. In this way each element of the matrix is between 0 and 1, but there is no guarantee that each row of the matrix selects only one point. To accomplish this constraint one hot and argmax operation should be applied; in this way the argmax returns the position of the highest value in each row of the matrix. The one hot instead creates a vector full of zeros, except in the position returned by argmax:

$$S = one\_hot(argmax(\hat{S})) \tag{4.14}$$

These two operations unfortunately are not differentiable, so it is proposed an approximation based on softmax and a temperature parameters like in the projection operation of Chamfer distance:

$$\hat{S} = softmax(\hat{S}/\tau) \tag{4.15}$$

When $\tau \to 0$, each column in $\hat{S}$ degenerates into a one hot distribution such S. During the training phase (4.15) it is applied so that the total loss can be backpropagate; instead, during inference (4.14) it is used because loss is not computed and backpropagated.

To further boost the performance of the network, a random relaxation is introduced; this consists of adding to $\hat{S}$ a random matrix defined as:

$$\hat{S} = \hat{S} + Random(\gamma) \quad where \quad Random(\gamma) \in R^{m \times n} \tag{4.16}$$

γ is the upper bound of the random number generated, and its value is computed as:

$$\gamma = r^{\frac{current\_step}{decay\_step}} \quad with \quad r \in [0, 1] \tag{4.17}$$

In this way, the random matrix is decayed exponentially and it tends to 0.

4.2.2 Loss

The sampling matrix generated in this way does not generate unique points; in fact, the only constraint is on the row of the matrix to be one hot, but a single point can be selected multiple times, thus leading to a high computational cost. To address this issue, a sampling loss is proposed in [10] to mitigate this phenomenon and increase the number of unique points. This loss accumulates each value of a column, which represents whether or not the original point is selected. The ideal case is that a point is selected zero or once. After this step, 0.5 is deducted and its absolute value is computed. Then we subtract 0.5 again and accumulate the loss in the other dimension. Lastly the absolute value is divided by the number of points N. This loss encourages element of each column to be 0 or 1 because if a

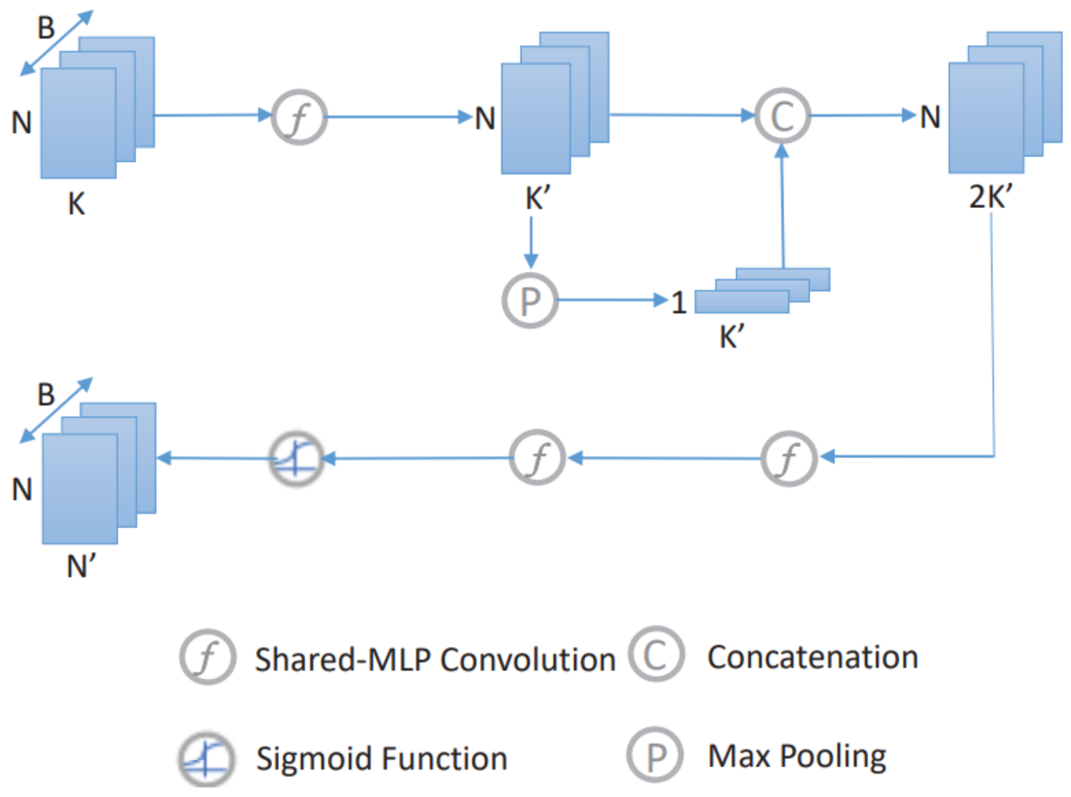

Figure 4.3: LSNet - Scheme of the network proposed to construct the selection matrix. Source [10].

single point is selected more than once, the first accumulation results in a value > 1 that cannot be compensated with the two subtraction of 0.5.

$$\mathcal{L}_{sample} = \frac{1}{N}\sum_{j=1}^{N}\left(\left|\left|\sum_{i=1}^{M}\hat{S}[i,j] - 0.5\right| - 0.5\right|\right) \tag{4.18}$$

Chapter 5

Introduction to Datasets

Datasets are a fundamental component of artificial intelligence. It is of paramount importance to ensure that the data are as clean as possible, devoid of outliers, and organized in a thoughtful manner. Additionally, the volume of data is crucial, especially in the context of deep learning, where the neural network's parameter count is exceedingly high, necessitating a substantial amount of data to mitigate overfitting. Other essential operations in dataset management include data pre-processing and augmentation.

5.1 Relevant Operations in Datasets

Working with datasets involves several critical operations to ensure data are suitable for training and evaluating artificial intelligence models. Two of the most significant operations are data pre-processing and data augmentation.

5.1.1 Pre-processing

Data pre-processing is aimed at preparing data for use in a neural network. It involves activities such as the removal of outliers, normalization, and feature engineering. The removal of outliers is essential to ensure that the data remain consistent and representative of the problem at hand. Normalization ensures that data features are on the same scale, enabling the machine learning model to learn effectively. Feature engineering encompasses the creation of new variables or features from existing data to enhance model performance.

5.1.2 Augmentation

Data augmentation is an operation that involves applying transformations to data to increase the number of samples. This is particularly valuable when dealing with limited datasets or when seeking to enhance a model's robustness. Typical transformations include rotation, translation, the addition of Gaussian noise, and other modifications that diversify the dataset.

In summary, the use of clean, well-organized, and appropriately pre-processed datasets is fundamental to building effective artificial intelligence models. A well-prepared dataset serves as the foundation upon which machine learning algorithms and deep learning models are constructed, enabling them to competently address a wide array of complex tasks.

5.2 Dex-Net 3.0 as a Dataset

5.2.1 What is Dex-Net 3.0?

Dex-Net 3.0[1] is a collection of data and tools designed to train and evaluate artificial neural networks in the context of object grasping. It has been developed by the Autonomous Systems Lab at the University of California, Berkeley, and is a significant contribution to the robotics and machine learning community.

The Dex-Net 3.0 dataset is an extension of the previous versions of Dex-Net and contains a wide range of data collected from simulations and real robots. These data have been used to train and evaluate object grasping algorithms, making Dex-Net 3.0 a vital tool for research in robotics and artificial intelligence.

5.2.2 Key Features of Dex-Net 3.0

Dex-Net 3.0 has several notable features that make it a reference dataset for object grasping. Some of its main features include:

- Object Diversity: The dataset contains a diverse set of real 3D objects, each with different shapes, sizes, and materials. This diversity better reflects the real-world challenges that robots face when grasping objects in unstructured environments.

- Pose Diversity: The data includes various poses of objects, simulating real-world situations where objects are arranged differently, forcing robots to adapt to various conditions.

- Point Cloud Size: One notable feature of Dex-Net 3.0 is that the point cloud data is 32x32.

- Labels: An interesting aspect of the dataset is that only one of the points in each 32x32 point cloud is labeled, highlighting a potential defect or challenge in the grasping task.

5.2.3 Dex-Net 3.0 in Detail

Dex-Net 3.0 is a valuable dataset of point clouds, particularly significant in the context of vacuum gripper-based robotic manipulation. It distinguishes itself as one of the few datasets tailored to assess grasp points in the context of vacuum grippers. The dataset is composed of 55 objects from various categories, including tools, groceries, office supplies, toys, and 3D printed industrial parts. These objects are further categorized into three groups based on their complexity: **Basic**, **Typical**, and **Adversarial**. The Basic category consists of 25 prismatic objects, the Typical category includes objects with planar surfaces of varying sizes (25 objects), and the Adversarial category contains objects with complex geometries that are challenging to grasp (5 objects).

To create this dataset, these 55 objects are placed in different poses, resulting in the generation of a remarkable 2.8 million point clouds. Each of these point clouds represents a view of a scene composed of a single object. However, what sets Dex-Net 3.0 apart is that, in these views, only the central point has a label that measures the probability of a successful grasp. This labeling approach adds a unique challenge to the dataset, as it introduces an element of uncertainty regarding the optimal grasp point. It is worth noting that the size of the point cloud, which is 32x32, is a critical factor in determining the potential grasp point.

The complexity and diversity of objects, categorized based on their difficulty, along with the varied poses, make Dex-Net 3.0 a versatile and challenging resource for the development and evaluation of object grasping algorithms.

In summary, Dex-Net 3.0 is a dataset of great significance that has revolutionized research in object grasping. Its diversity of objects and poses along with the specific characteristics of the point cloud data and the labeling, make it an essential tool for the development of robots capable of manipulating objects in complex environments. This dataset represents a significant step towards advancing robotic autonomy in grasping objects.

5.3 SuctionNet 1 Billion as a Dataset

5.3.1 What is SuctionNet 1 Billion?

SuctionNet 1 Billion[2] is a comprehensive dataset compiled to train and evaluate machine learning models used in robotic systems that employ suction-based manipulation techniques. This dataset has been created by professors and students from Shanghai Jiao Tong University and it stands as a substantial resource for the robotics and artificial intelligence communities.

The dataset comprises an astonishing one billion data samples, making it one of the largest and most diverse datasets ever assembled for robotic manipulation.

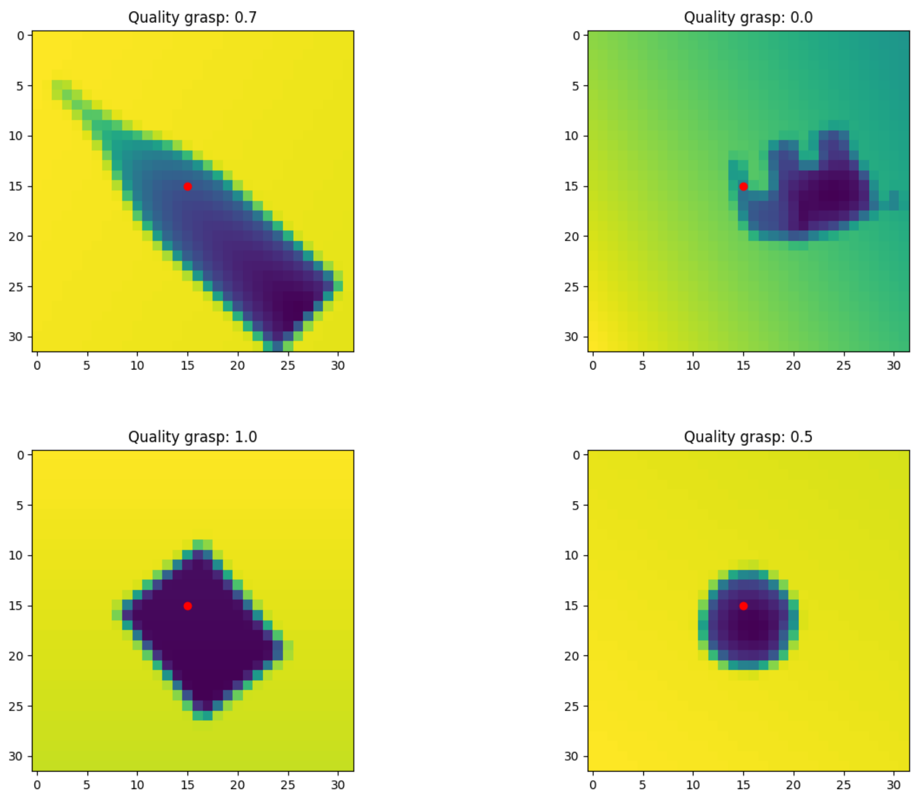

Figure 5.1: Example of depth image contained in DexNet3.0. The first one is a view of a bottle and on the top of the image can be seen the quality of the grasp, that is the red point in the image. The second view is a toy elephant. The third one is a box and the last is a ball.

What sets SuctionNet 1 Billion apart is its representation of real-world scenarios where scenes consist of multiple diverse objects, each presenting unique challenges for suction-based grasping.

5.3.2 Key Features of SuctionNet 1 Billion

SuctionNet 1 Billion boasts several notable features that make it a critical resource for advancing the field of robotic suction-based manipulation:

Massive Scale

The dataset contains one billion data samples, providing an unprecedented volume of data for training and testing machine learning models. This extensive scale allows for robust model development and evaluation.

Object Diversity

SuctionNet 1 Billion includes a vast array of objects, encompassing different shapes, sizes, materials, and surface textures. This diversity reflects the complexity of real-world scenarios and challenges that robotic systems must address.

Scene Composition

One of the distinguishing characteristics of SuctionNet 1 Billion is its representation of scenes composed of multiple diverse objects. This scene complexity closely mimics real-world environments, making the dataset particularly relevant for practical robotic applications.

Data Split: Train, Similar, Novel

SuctionNet 1 Billion introduces a data split into three categories: **train**, **seen**, **similar**, and **novel**. In the 'train' category, there are objects that are used for training and have a significant presence in the dataset. The 'seen' category is an extension of train set. The 'similar' category contains objects that are somewhat similar to the training objects, introducing a level of transfer learning. The 'novel' category presents the most challenging scenario, as it includes objects that are not present in the training data. This division facilitates the assessment of the model's adaptability to novel objects, a critical capability for real-world robotic applications.

5.3.3 SuctionNet 1 billion in Detail

SuctionNet 1 Billion offers a detailed view of its composition, providing insight into the dataset's structure and characteristics.

The dataset comprises a total of 190 scenes, divided into distinct categories to support different training and evaluation scenarios. Specifically, it includes 100 scenes designated for the 'train', category, 30 scenes for 'seen' object, 30 scenes for 'similar' and 30 scenes for 'novel'. Each of these scenes encompasses 256 unique views of the same environment, enabling comprehensive coverage of the object interactions within these scenes.

One notable feature distinguishing SuctionNet 1 Billion from its predecessors, such as Dex-Net 3.0, is the labeling complexity. For each object in a scene, multiple labels are provided, reflecting the diverse grasping possibilities. This fine-grained labeling offers a more in-depth understanding of object interaction dynamics and enhances the dataset's utility for training advanced robotic systems.

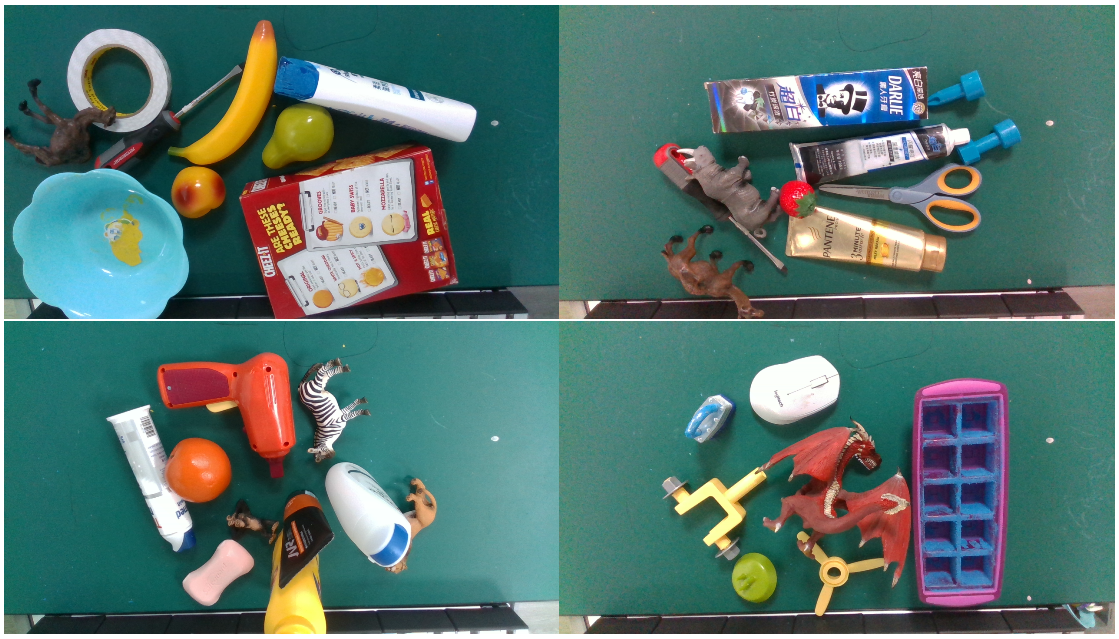

Figure 5.2: Example of views of the scene contained in SuctionNet from different data split. Respectively: train, seen, similar and novel set

Furthermore, it is important to note that the scenes in SuctionNet 1 Billion are captured from the physical world using RealSense cameras. While this real-world data add a layer of authenticity, they also introduce challenges. The scenes may contain inherent noise and variations due to real-world conditions, making the dataset a more accurate representation of the complexities encountered in actual robotic manipulation scenarios.

This authenticity, while challenging for training, is essential for the development of robotic systems that must operate effectively in unstructured and dynamic environments. SuctionNet 1 Billion's real-world scenes and labeling make it a valuable resource for advancing the field of robotic suction-based manipulation, bridging the gap between simulation and real-world application.

5.4 SuctionNetCAD: CAD Models Within SuctionNet

5.4.1 What is SuctionNetCAD?

SuctionNetCAD is not a new dataset, it is extrapolated by SuctionNet. In fact, with the dataset SuctionNet the authors release also the CAD model of every single object present in the dataset.

5.4.2 Key Features of SuctionNetCAD

SuctionNetCAD introduces several key features that enrich the SuctionNet dataset:

CAD-Based Point Clouds

SuctionNetCAD integrates CAD models of objects into the SuctionNet dataset. This addition allows for the generation of precise and structured point clouds based on the CAD geometries of the objects, resulting in a dataset that offers the best of both the virtual and real worlds.

Comprehensive Point Labeling

In the spirit of SuctionNet, SuctionNetCAD retains the practice of comprehensive labeling, ensuring that every point in the point clouds is labeled. This fine-grained labeling provides detailed information about the graspability of each point, contributing to the continued refinement of grasping strategies.

Noise-Free Data

A noteworthy characteristic of SuctionNetCAD is the absence of noise in the point clouds. By eliminating noise from the dataset, it creates an idealized environment, serving as a valuable resource for testing and validating robotic grasping algorithms.

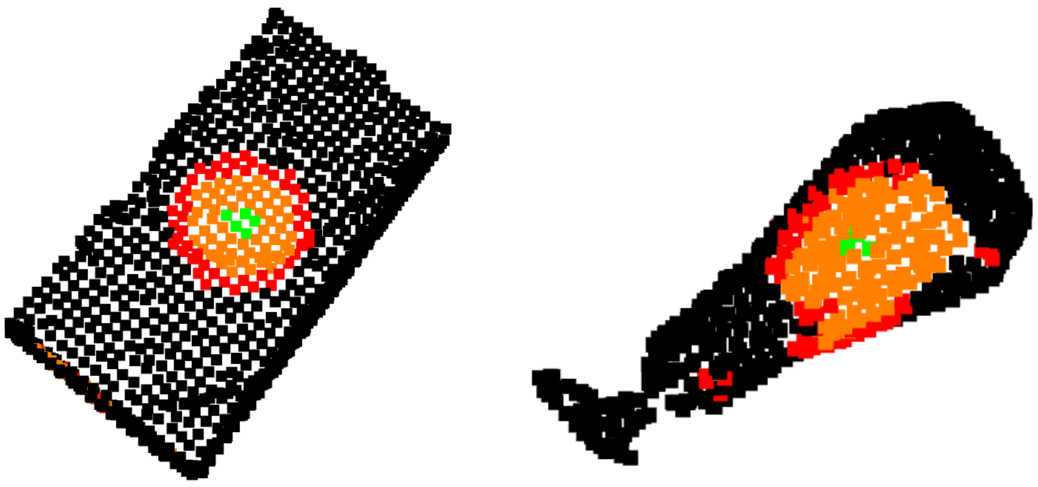

Figure 5.3: Example of views of the objects contained in SuctionNetCAD. Green points are the best, the orange and red ones gradually have lower scores. Black points are the ones under the selected threshold.

Chapter 6

Proposed Network

In this work, we propose two approaches to accomplish the task. The first is based on the classification of point clouds. The latter, instead, aims at subsampling the points for grasping through a sampler and then classify these points to obtain scores for all the extracted points. Therefore, the second approach is an extension of the first, which leverages the sampler to simplify the point cloud. In the following section, we delve into the details of how these two networks are constructed and how we execute the training phase.

6.1 Classification task

The goal of the classification task is to learn the optimal points for grasping on the object's surface. In this context, the network outputs a binary value that indicates whether a point is suitable for grasping or not, with this value falling within the range [0, 1]. A value exceeding a predetermined threshold is considered positive for grasping, while a value below the threshold is deemed unsuitable. This threshold is selected based on the dataset and may vary among datasets.

6.1.1 Classification on DexNet3.0

The first dataset we use to train our network is DexNet3.0, as mentioned in the previous chapter. This dataset consists of only one labeled point in the point cloud, requiring the network to classify a single point. The architecture of our network is constructed based on PointNet, a neural network specifically designed for processing point clouds.

The network architecture can be divided into three main components: the feature extractor, aggregator, and classifier.

During the feature extraction phase, the network takes a three-dimensional point

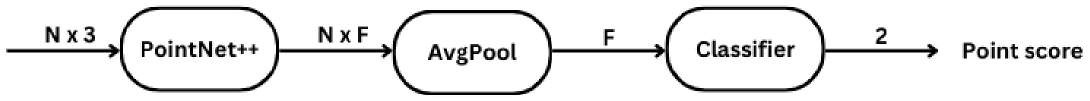

Figure 6.1: Network classification

cloud as input. PointNet processes each point individually to learn distinctive features relevant to the classification task. This transforms the raw point cloud data into a higher-level feature representation.

$$f : \mathbb{R}^{N\times 3} \rightarrow \mathbb{R}^{N\times F} \tag{6.1}$$

where f is the feature extractor function, N is the number of points in the cloud, and F is the number of extracted features per point.

Following the feature extractor stage, we employ an average pooling operation to aggregate the per-point features. The aggregator computes the average for each feature across all the points, resulting in a global feature representation:

$$g : \mathbb{R}^{N\times F} \rightarrow \mathbb{R}^{F} \tag{6.2}$$

where g is the aggregation function, and F represents the dimension of the aggregated feature representation.

The final component of the network is the classification module. It consists of fully connected layers that take the aggregated features as input and produce a binary output, indicating the probability of the point belonging to a specific class:

$$h : \mathbb{R}^{F} \rightarrow \mathbb{R}^{2} \tag{6.3}$$

Function h is the classifier, and its output represents the predicted class probabilities.

Enhancements

To further improve the network's performance, we introduce an extension. Firstly, we incorporate a one-hot vector to indicate the point to be evaluated within the point cloud, similar to the approach used in GQCNN, where the grasp point is passed as input. Instead of directly passing the point, we indicate its position within the point cloud.

Additionally, we extend the approach by inserting ones in the neighborhood of the selected point. This modification allows the network to consider local context when making classifications.

Loss function

The loss function used to train the network is the Cross-Entropy Loss, which measures the difference between the predicted probability distribution and the ground truth labels. This loss function is commonly employed in classification tasks:

$$L(y, p) = -(y \log p + (1 - y) \log(1 - p)) \tag{6.4}$$

where L is the cross-entropy loss, y represents the ground truth labels from the dataset, and p corresponds to the network's predictions.

Training details

The network is trained with a batch size of 256, and the training process is conducted using Google Colab.

6.1.2 Classification on SuctionNetCAD

In this section, we present a description of the neural network architecture employed in our research, specifically tailored for the SuctionNetCAD dataset. This network is an extension of the principles established in our previous architecture, initially designed for the classification of a single point within a three-dimensional point cloud. We have made adjustments to suit the requirements of the SuctionNetCAD dataset.

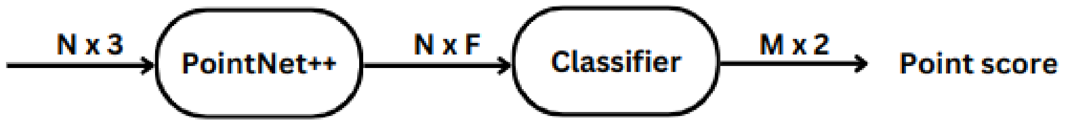

Figure 6.2: Network for classification of all points

The network architecture is organized into two primary components: feature extractor and classifier.

Feature extraction

The modification to the architecture primarily affects the backbone architecture. PointNet++ has been introduced as backbone architecture for this network, considering the increased complexity of the classification task. PointNet++ is capable of capturing intricate point interactions due to its hierarchical structure, which is essential for classifying multiple points within a complex point cloud.

PointNet++ is organized into two hierarchical PointNet modules with a Multi-Stage Grouping approach. In the first module, we perform the subsampling of 2048 points using the Farthest Point Sampling (FPS) algorithm. Subsequently, these selected points are subjected to ball queries with radii of 0.025. The goal is to sample at least 32 and 64 points for the respective radii.

The second PointNet module subsamples 1024 points and performs ball queries with radii of 0.05. We sample 32 and 64 points, respectively. The hierarchical structure allows our network to capture complex point interactions at different scales, contributing to robust classification.

The input and the output of the network remains the same of the previous approach.

$$f : \mathbb{R}^{N\times 3} \rightarrow \mathbb{R}^{N\times F} \tag{6.5}$$

Here, f represents the feature extraction function, N is the number of points in the cloud, and F denotes the number of extracted features per point.

Feature aggregation

In contrast to the previous network, we have omitted the feature aggregation step in this architecture, as we aim at classifying all the points individually rather than aggregating them. Therefore, the features extracted in the encoder are passed directly to the classifier.

Classification

The classification component remains vital to our network, where convolutional layers are employed to process the extracted features and produce classification outputs for all points within the point cloud:

$$h : \mathbb{R}^{N\times F} \rightarrow \mathbb{R}^{N\times 2} \tag{6.6}$$

The function h is responsible for classification, and the output represents the predicted class probabilities for each point.

The classifier has 3 convolutional layers, each complemented by batch normalization [11] and Rectified Linear Unit (ReLU) activation functions [12]. This architecture is designed to process relevant features from high-dimensional inputs and produce the final classification output.

Loss function

To train the network we employ two different losses: Cross Entropy and Mean Squared Error. The first one is the classification loss (6.4) that is the same of the previous network. The last loss is used to further boost the precision of the best point suitable for grasping: we employ a Mean Squared Error between the best point from the label and the best point given as output from the network:

$$MSE = \frac{1}{M}\sum_{i=0}^{M}(p_i - p_i^*)^2 \tag{6.7}$$

In this case M is equal to 5, p_i are the best M points predicted from the network and p_i^* is the best point from the label.

Training Details

The network is trained with a batch size of 8, and the training process is conducted using Nvidia Quadro M4000. The learning rate is set to $1e^{-3}$ and no weight decay is used.

6.1.3 Improvements

To enhance the network's performance, we employ techniques outlined in [13]. These improvements involve integrating an encoder before feeding the point cloud into PointNet++. This modification enables the backbone to extract features from a high-dimensional space. Another strategy detailed in the paper involves introducing regularization after the grouping operation. In fact, PointNet++ computes the relative position of points within the grouping radius with respect to the centroid. This distance depends on the radius size and is smaller than it, requiring the network to learn larger weights for optimization. This regularization is crucial, ensuring that the features remain within a fixed range. Another improvements we found that helps the network to learn and generalize is concatenate to the point clouds also the information of the normals in that point. The normals provide information about how the object is placed in space and how the robot should arrive to that point.

6.2 Regression task

The regression task aims at learning the optimal points for grasping on the object surface. This is achieved through a sampler that outputs a subsample of the original point cloud containing the relevant points. In this chapter, we assess two distinct samplers, as described in Chapter 4. The points extracted by the sampler are subsequently classified to provide a binary score for each point. The overall network is composed of a feature encoder, a sampler, and lastly, a classifier.

6.2.1 Regression with selection matrix

In this section, we introduce an advanced network architecture tailored to the SuctionNet dataset. The enhanced architecture incorporates PointNet++ and a sampler to effectively handle complex point cloud data and perform both regression and classification tasks.

The network architecture builds upon the principles of our previous design, has three main components: feature extraction, sampler, and classifier.

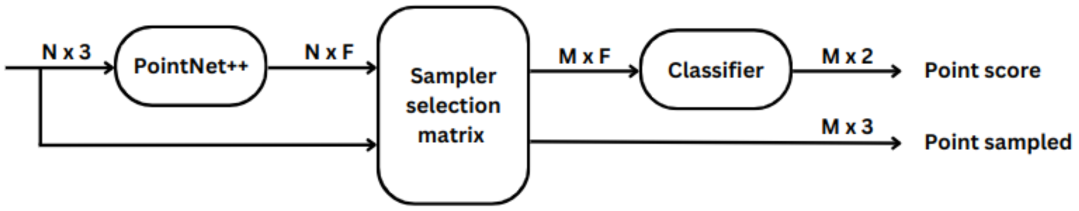

Figure 6.3: Network with selection matrix

Feature extraction

The feature extraction stage remains consistent with PointNet++. The feature extraction function is defined as (6.5).

The backbone architecture, PointNet++, plays a crucial role in the network's capability to handle complex point cloud data. PointNet++ is organized into three hierarchical Set Abstraction stages, employing Multi-Stage Grouping to refine point extraction.

In the first stage, 4096 points are subsampled, and two ball queries are applied with radii of 0.05 and 0.1, resulting in the selection of 16 and 32 points, respectively. Subsequent stages continue to refine the point extraction process, with 2048 and 1024 points subsampled using Farthest Point Sampling (FPS). The radii for ball queries are doubled, and the number of points for the ball queries remains consistent across all stages, allowing for effective point extraction at varying scales.

Sampler

A sampler has been introduced in this architecture, based on a selection matrix described in Section 4.2. This matrix is employed to extract points from the point cloud, thereby reducing the overall dimensionality of both the point cloud and the extracted features. The sampler takes per-point features as input, and outputs a matrix that indicates which points need to be selected. This matrix is then utilized to compute the reduced point cloud and features. Figure 6.4 provides a schematic representation of these operations. The overall function is as follows:

$$g : \mathbb{R}^{N\times a} \rightarrow \mathbb{R}^{M\times a} \tag{6.8}$$

where a is 3 or F.

The sampler has the same number of layers of the ones described in Section 4.2. The selection matrix created by the sampler has dimensions of (N, 256), which allows the effective selection of points from the original point cloud. The Batch normalization [11] and ReLU activation [12] function are inserted after each convolutional layer to prevent over-fitting and add non-linearity to the network.

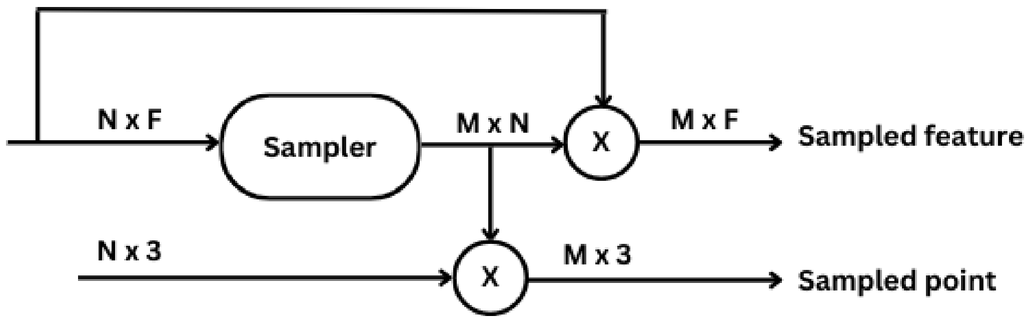

Figure 6.4: Schematic representation of the operation used to compute the extracted points and features from the sampling matrix. N x F are the feature from the PointNet++ module, N x 3 is the point cloud

Classifier

The classifier component has been retained with the same number of layers, ensuring the effectiveness of the architecture. The classifier processes the reduced-dimensional features and classifies the points based on the information provided by PointNet++ and the sampler.

Loss

The losses used to train the network are three, one for classification and two relative to the sampler and the point extracted. The first is the already mentioned Cross-Entropy loss (6.4) used for the classification task. The second loss is given by the Mean Squared Error (6.7) between the sampled points and the labels; this loss helps the sampler to extract relevant points of the object:

$$\mathcal{L}_{regression} = \frac{1}{M}\sum_{i=0}^{M}(p_i - p_i^*)^2 \tag{6.9}$$

where p_i are the points extracted from the sampler, p_i^* are the points of labels and M are the number of points extracted (in this case 256).

The last loss is the one described in Section 4.2.2 to ensure that the matrix is discrete and the points are spread across the object surface.

Training details

The training process is performed with batch size of 4, an SGD optimizer with learning rate of $l_r = 1^{-3}$ and weight decay $w_d = 1^{-3}$. The temperature parameter τ used in the sampler is equal to 0,033. All the training phase is performed on Nvidia Quadro M4000.

6.2.2 Regression with Chamfer distance

In this section, we propose a second approach to subsample points of point cloud. This network shares the backbone and the dataset with the previous network, as we want to make a comparison between the two approaches. The network we use is similar to the one described in [5], we adapt that code to our task. In fact, they use a parallel jaw gripper so they have a module that for each point of grasp regress the second contact point and also the pitch angle. We do not need this information because we use a vacuum gripper, so we remove that module.

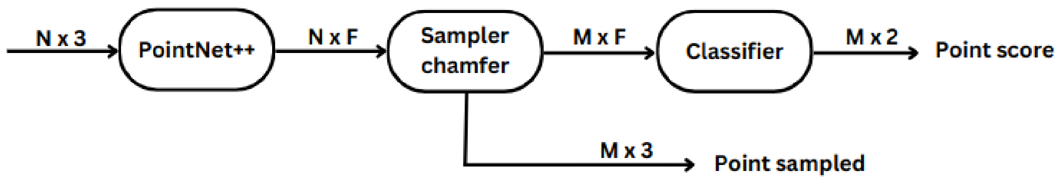

Figure 6.5: Network with chamfer distance

Sampler

The sampler introduced in this network is the one described in Section 4.1. The module learns what are the points most suitable for grasping through a minimization of the Chamfer distance (4.2). The sampler takes the per-point feature as input and returns the simplified point cloud with the feature of each points like the previous one. The sampler is made up by 4 fully connected layers with Batch Normalization [11] and ReLU [12].

Loss

The loss used to train the network are: Binary Cross Entropy for the classification part, Mean Squared Error on the extracted points and lastly the two losses described in Section 4.1 that are Chamfer distance (4.2) and projection loss (4.6).

Training details

The parameters used for training are the same of the ones adopted in the reference paper [5]: $l_r = 1^{-4}$, $w_d = 1^{-4}$, batch size of 1, number of sampled grasp equal to 500

Chapter 7

Experiments

7.1 Metrics

To evaluate the performance of the network built we propose different metrics to make comparison among them and understand what is the best approach and in which conditions.

7.1.1 Metrics for classification

For the classification task we use different metrics: the accuracy, precision and recall. These metrics are useful to understand how good the network is in classification. We also use a distance metrics to evaluate how close is the best point from the network to the real best point.

Accuracy

Accuracy is a measure that expresses how well the network can correctly classify examples in the test dataset. In other words, it represents the percentage of cases where the network has produced the correct prediction compared to the total number of cases tested.

Accuracy is computed as:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total test cases}} \times 100 \tag{7.1}$$

where 'number of Correct Predictions' is the number of cases in the test set where the network has made a correct prediction compared to the true class. The 'total test cases' represents the overall number of examples in the test set.

Accuracy provides an overall view of the neural network's performance. However, in presence of imbalanced classes (when some classes are more frequent than others in the dataset), accuracy may not be sufficient. In such situations, it is useful to examine additional metrics, such as precision, recall, or F1-score, which provide more detailed information about the network's ability to correctly classify different classes.

Precision

Precision is a metric that measures the accuracy of the positive predictions made by the network. It specifically quantifies the percentage of instances predicted as positive by the network that are actually true positives.

Precision is computed as:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \times 100 \tag{7.2}$$

where 'true positives' are the cases in the test set where the network correctly predicted the positive class; 'false positives' are the cases where the network incorrectly predicted the positive class.

Precision is particularly useful in situations where the cost of false positives is high.

Recall

Recall is a metric that measures the ability of the network to capture all the relevant instances of a positive class. It quantifies the percentage of true positive instances that were correctly identified by the network.

Recall is computed as:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \times 100 \tag{7.3}$$

where 'true positives' are the cases in the test set where the network correctly predicted the positive class; 'false negatives' are the cases where the network failed to predict the positive class when it was actually positive.

Recall is particularly important when the cost of false negatives is high.

F1 score

The F1 score is a metric that combines both precision and recall into a single value, providing a balanced measure of a classifier's performance. It is particularly useful when there is an uneven class distribution.

F1 score is computed as the harmonic mean of precision and recall:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{7.4}$$

The F1 score ranges from 0 to 1, where 1 indicates perfect precision and recall, and 0 indicates the worst possible performance.

The F1 score is valuable in situations where false positives and false negatives carry different costs. It helps strike a balance between precision and recall, providing a comprehensive measure of a model's ability to correctly classify positive instances, while minimizing false positives and false negatives.

Euclidean distance

The Euclidean distance between two points in a three-dimensional space is a measure of the straight-line distance between them. It is commonly used to calculate the distance between predicted and true points in regression tasks.

For two points $P(x_1, y_1, z_1)$ and $Q(x_2, y_2, z_2)$, the Euclidean distance (d) is given by:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

The Euclidean distance provides a direct measure of how far apart two points are in space. In the context of machine learning, it is often used to evaluate the accuracy of regression models by comparing predicted and true values.

7.1.2 Metrics for evaluation of samplers

In the regression task we use different metrics to evaluate our network: the coverage of the grasping points present in the dataset, the number of unique points extracted from the sampler and the mean absolute error between the extracted points and the labels.

Coverage of grasping points

Coverage measures the percentage of predictions that fall within a specified threshold, often denoted as ϵ. This metric is particularly useful when assessing the model's accuracy in predicting values within a certain acceptable range.

Coverage is computed as:

$$\text{Coverage} = \frac{\text{Number of predictions within } \epsilon}{\text{Total number of predictions}} \times 100 \tag{7.5}$$

where Number of predictions within ϵ is the count of predictions that fall within the specified threshold ϵ; total number of predictions is the total number of predictions made by the model.

This metric helps to quantify the model's ability to make accurate predictions within a given tolerance, in your case, within a 2.5 cm threshold.

Unique points

Unique points measures the percentage of points that are unique out of all those drawn. This metrics is particular useful when we deal with sampler based on a neural network because they have the tendency to extract multiple times the same points.

The percentage of unique points is computed as:

$$\text{Unique points} = \frac{\text{Number of unique points}}{\text{Number of total point extracted}} \times 100 \tag{7.6}$$

This metrics quantifies the model's ability to extract different points.

Mean absolute error

In the context of regression tasks, Mean Absolute Error (MAE) is a metric that measures the average absolute difference between the predicted values and the true values. It provides a straightforward measure of the model's accuracy.

MAE is computed as:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\text{Predicted}_i - \text{True}_i| \tag{7.7}$$

where n is the total number of examples in the dataset. Predicted_i is the predicted value for the i-th example. True_i is the true value for the i-th example.

MAE is expressed in the same unit as the target variable, and represents the average magnitude of the errors between predicted and true values. It is less sensitive to outliers compared to other error metrics, like Mean Squared Error (MSE).

7.2 Result

In this section for each Network trained we describe what are the metrics used to evaluate it with the relative value. We also make a comparison between other networks, when it is possible.

7.2.1 Classification on DexNet3.0

To evaluate this network we use **accuracy**, **precision**, **recall** and **F1 score**. We use these four metrics because the dataset is imbalanced with respect to the positive class (the points suitable for grasping). We compare this network with GQCNN [1], that is the network proposed by the creator of the dataset. The authors do not

Networks	Accuracy	Precision	Recall	F1 score
GQCNN	93.5%	-	-	-
Ours	87.9%	84.7%	92.8%	88.6%

Table 7.1: Table resuming the metrics used to evaluate the two networks. It can be seen how the overall accuracy is similar between the two approaches.

publish the value for precision and recall, so we can not make a perfect comparison between them but it is still useful to do so.

As evidenced in Table 7.1 out network is slightly worse than the GQCNN although we don't know the other metrics and how it performs. The two approaches are also quite different: ours network uses point clouds, GQCNN uses depth images. The networks used to process point clouds are quite new, instead the ones used to process images are better known and explored.

7.2.2 Classification on SuctionNetCAD

The evaluation on the SuctionNetCAD dataset is instead performed as in the previous approach, but we add a metrics to evaluate how close is the best prediction of the network to the effective best label, as described in Section 7.1.1. We compute the distance for the top 1 and top 5 of the prediction. With this dataset we do not have a network to make a comparison, because the dataset is not used in any network for how it's made.

Networks	Accuracy	Precision	Recall	F1 score	Top 1	Top 5
Ours	89.6%	82.4%	55.1%	65.9%	0.7 cm	0.3 cm

Table 7.2: Table resuming the metrics used to evaluate the network

The most relevant metrics are the two computed on the top 1 and top 5 prediction, because we can see how close are the best points predicted from the network to the truth best point. This means that the points extracted are aligned well to the label present in the dataset. The poor recall is due to our choice to have few positive prediction near the best label, and these points with higher confidence to be real positive. In the Figures (7.1), (7.2), (7.3), (7.4) we see some prediction of our network tested with object that belongs to **Similar** section of SuctionNetCAD.

As we can see in Figure 7.1, the points classified as suitable effectively overlap with the labels present in the dataset. This demonstrates that the network has successfully learned how to classify points correctly. Naturally, there are false positives and false negatives, as expected and as indicated in the results Table 7.2, but the overall outcome is satisfactory. Regarding the top 5 predictions, we

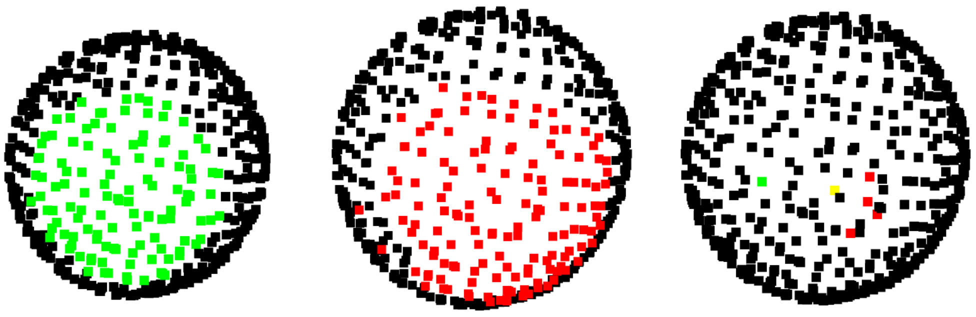

Figure 7.1: The first image represent a point cloud of a ball with in green the points suitable for grasping. The second image represent the prediction of our network, in red the points with a prediction larger than the threshold. In the last image there are the best point from the label (in green) and the top 5 prediction of the network (in red), in yellow there is the best point classified by the network.

can observe that they are among the closest to the labeled optimal point and are generally positioned centrally. Given the nature of the considered object, this positioning undoubtedly makes the grasp feasible and accurate.

Also in the case in Figure 7.2 the network performs well, the predictions are well overlapped to the labels and the best prediction is near the optimal points from the ground truth.

These two objects are two simple case in fact ball and boxes are quite simple to grasp and learn what are the best surface to do so. They are smooth and simmetric.

As evident from the Figure 7.3, in this instance, the network failed to predict the grasping area correctly (green points on the drill's handle). It correctly classified only a few points on the drill's head. Even when evaluating the best-extracted points, they appear to be a bit farther away compared to previous cases.

In this case Figure 7.4, we can observe that the grasping area defined in the dataset is not very centrally located with respect to the object and tends towards the left. Certainly, that area would also be suitable for grasping, given the shape of the object. However, in my opinion, the network's prediction is more centrally positioned and still accurate. This is an example of an object where, based on the labels in the dataset, the network did not perform well, leading to a decrease in the metrics in the table. However, with a visual and qualitative assessment, we can assert that this is not the case.

The network exhibits good performance, as we have seen, and generally avoids significant errors. Instances where it fails to classify any point as suitable are rare, and the distance between the top predictions and the ground truth is minimal. In the next paragraph, we will also explore how the network performs with a real robot and what its success rate is.

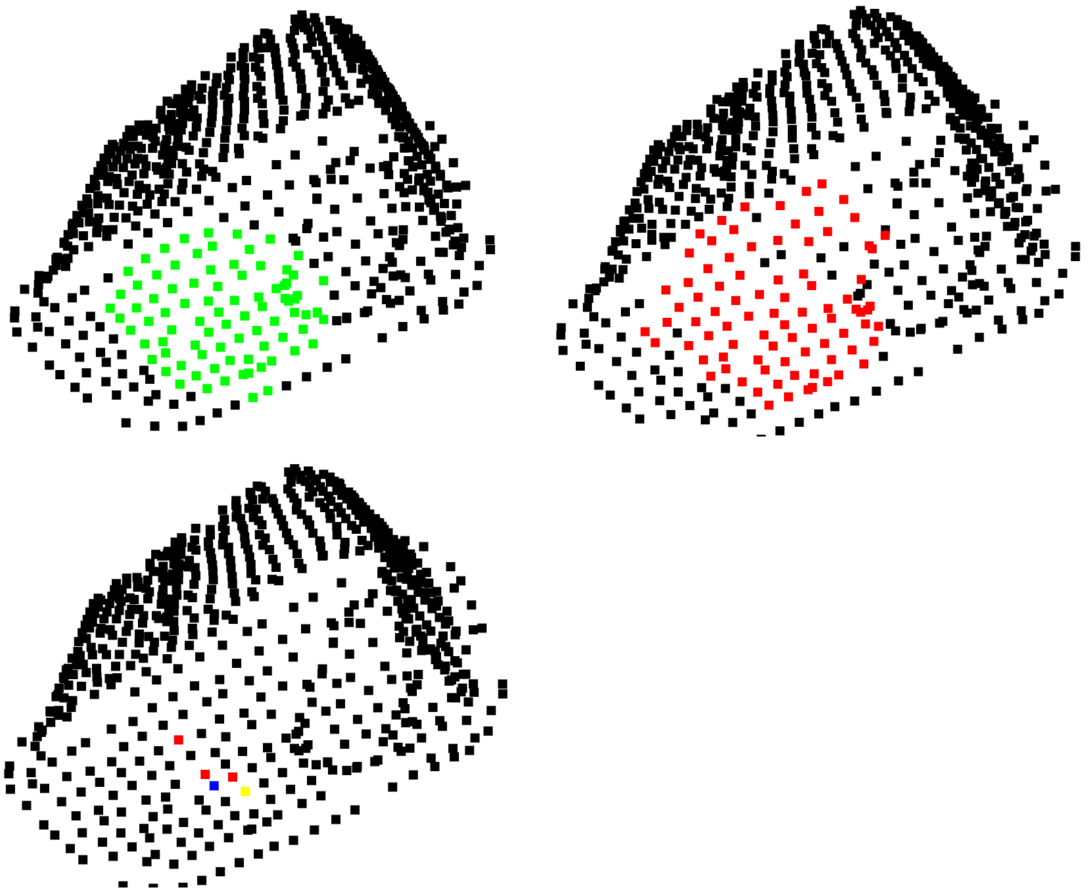

Figure 7.2: The first image represent a point cloud of a box with in green the points suitable for grasping. The second image represent the prediction of our network, in red the points with a prediction larger than the threshold. In the last image there are the best point from the label (in blue) and the top 5 prediction of the network (in red), in yellow there is the best point classified by the network.

7.2.3 Comparison between samplers

In this section we present the result from the two samplers and the metrics used to evaluate them. In particular the metrics used are the ones described in Section 7.1.2: coverage of points, unique points and MAE.

As it can be seen from the comparison the selection matrix has a worst coverage of the points with respect to the sampler with Chamfer distance, this is because the latter tends to form a globular shape of the extracted points so they are all concentrated around a certain point. The selection matrix sampler instead generates points spread all over the shape of the object this is useful because in

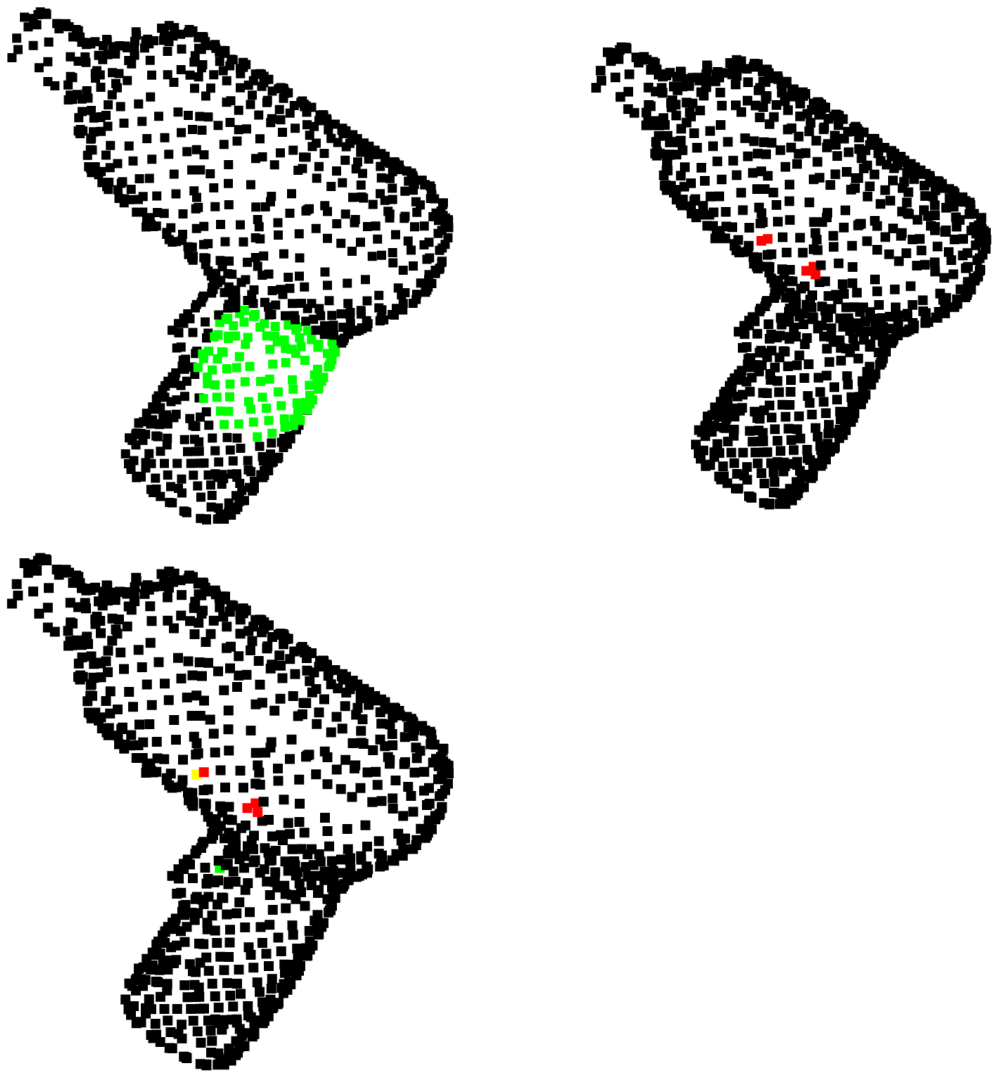

Figure 7.3: The first image represent a point cloud of a drill with in green the points suitable for grasping. The second image represent the prediction of our network, in red the points with a prediction larger than the threshold. In the last image there are the best point from the label (in green) and the top 5 prediction of the network (in red), in yellow there is the best point classified by the network.

this way if one point is wrong there are a lot of other points spread across the object that are probably good. Instead in the sampler a Chamfer distance if the center of the globular is not good, with high probability also the points around it are not good. The sampler with a Chamfer distance instead is more precise and has a lower MAE. We carried out this comparison by selecting points to extract in

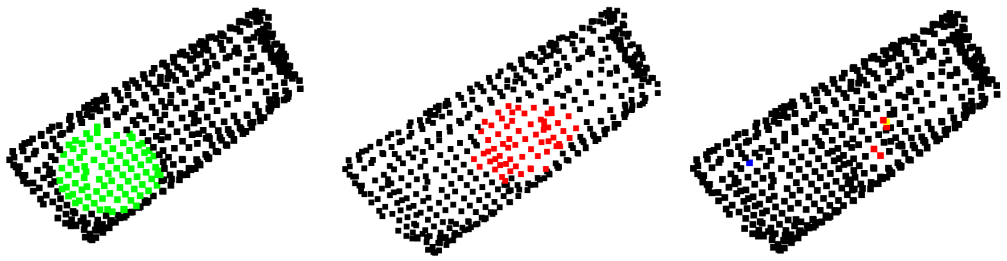

Figure 7.4: The first image represent a point cloud of a hand cream with in green the points suitable for grasping. The second image represent the prediction of our network, in red the points with a prediction larger than the threshold. In the last image there are the best point from the label (in green) and the top 5 prediction of the network (in red), in yellow there is the best point classified by the network.

Networks	Coverage	Unique points	MAE
Selection matrix	63%	77%	0.04
Chamfer distance	71%	45%	0.02

Table 7.3: Table resuming the metrics used to evaluate the two samplers.

such a way that the number of unique points is more or less equal. This ensures that the coverage metric is effectively comparable, as having a different number of unique points would have led to an advantage for one of the two.

Pros and cons

In this paragraph we see different comparison between the two sampler.

In the Image 7.5 we can see what said before and how the point extracted from the sampler based on a selection matrix are more spread across the object. We can also see how close are the points selected from the sampler with chamfer distance. Another important thing to point out is that the first sampler extract also noise points that are definitely not suitable for grasping, instead the second sampler is more robust to noise points and it typically doesn't extract them.

This last comparison Image 7.6, instead, shows a case in which the sampler that fails is the one based on the chamfer distance that extract points on the curvature of the object and consequentially all the points around it are not suitable.

In conclusion both samplers have strengths and weakness, each has cases in which fails and other in which is good.

In the next section we will see how the two different network performs with a real robot and what are the strengths and the weakness of the two networks.

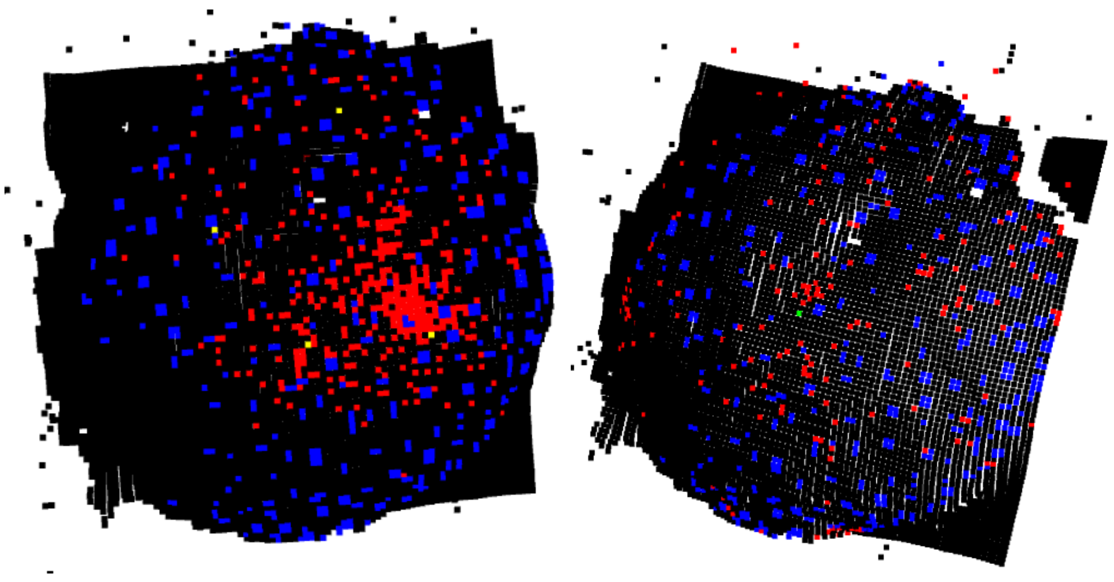

Figure 7.5: On the left there is a prediction of the network with the sampler with chamfer distance. On the right, instead, there is a prediction of the network with the sampler based on the selection matrix. The red points are the predicted ones, in blue are the labels from the dataset.

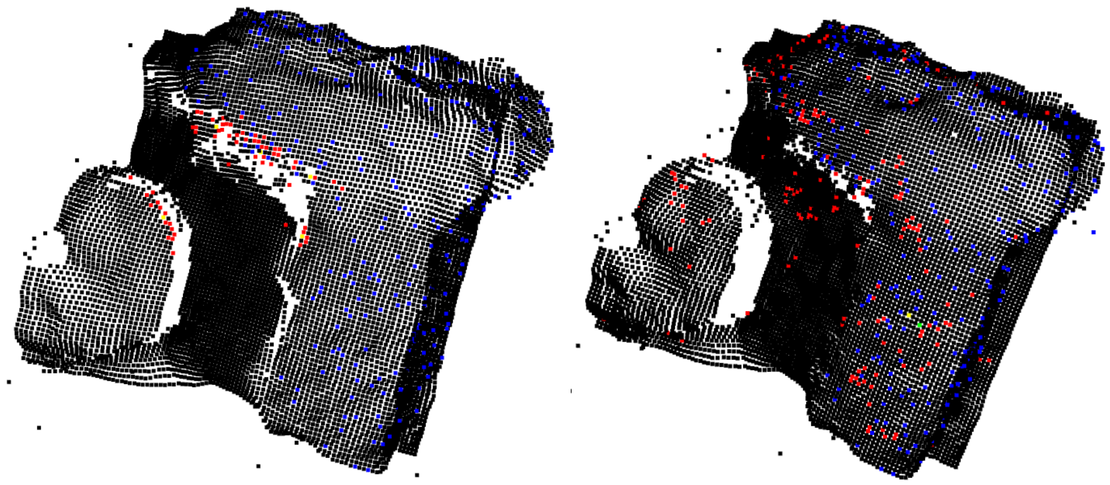

Figure 7.6: On the left there is a prediction of the network with the sampler with chamfer distance. On the right, instead, there is a prediction of the network with the sampler based on the selection matrix. The red points are the predicted ones, in blue are the labels from the dataset.

7.3 Real robot experiments

To evaluate the utilized neural network in more detail, tests were conducted with a real robot equipped with a suction gripper, specifically the Comau Racer 5 robot. The camera employed to capture the scene is an Intel RealSense, capable of acquiring both RGB and depth images, mounted on the robot's wrist. As for the gripper, it is from Schmalz, and it provides feedback on whether the vacuum has been successfully created, indicating the success or failure of the grasp. This feedback feature is particularly crucial as it allows for automating the task of determining whether the object has been grasped, eliminating the need to rely solely on visual inspection.

Regarding the testing methodology, objects were categorized into groups based on size, distinguishing between small and large objects. This initial categorization helps assess the network's ability to grasp objects of different sizes and the quality of these predictions. Smaller objects are generally more challenging to grasp due to having fewer viable grip points, and even a small error may prevent a successful grasp. Another category evaluated was transparent objects, posing a significant challenge because they are challenging for the camera to capture, resulting in point clouds with holes and noise, making the scene particularly difficult to process.

The robot's perspective involves a container containing the objects, one at a time, and the grasped objects are released into another container. The task is performed in a loop for each object until the predetermined number of grasps, set at 20, is reached. When released, the object falls onto a soft surface, causing it to bounce and land in varying positions, introducing randomness to the object's pose and allowing for the evaluation of multiple views of the same object.

In the event of a failed grasp, the algorithm is designed to continue making attempts using grip points with progressively lower predicted scores.

Big size objects

To conduct the tests, six large-sized objects were selected. These objects include a detergent bottle, a box, a ball, a jar, a bowl, and an unspecified object.

The Table 7.4 reports, for each object, the number of grasps that allowed the robot to successfully perform the task. Specifically, it can be observed that for large-sized objects, the success rate is quite high, indicating that the neural network generally has little difficulty predicting good grasp points. Upon analyzing the failed attempts, it becomes apparent that they occur when the object is in an unfavorable pose or when the normal direction of the chosen point is not perfectly consistent with reality due to noise present in the acquired image. Therefore, it might be necessary to review the algorithm used to estimate the normal or explore more suitable parameters.

Object	Success rate
Detergent	85%
Big box	85%
Big ball	60%
Jar	100%
Bowl	90%
Object 1	50%

Table 7.4: Table of success rate of the object evaluated. 20 trials were carried out for each object except for the object 1 in which 10 test were carried out.

Examining the table, it is evident that the object with the highest number of failed grasps is the ball. This is not because the network mispredicts the grasp point, in fact, the grasp point is accurately predicted at the center of the ball. However, the ball's seams prevent the suction cup from adhering perfectly to the surface and creating a vacuum. Excluding the ball from the objects, the success rate of the model for large-sized objects is 87%. Instead, considering it success rate decrease to 82%.

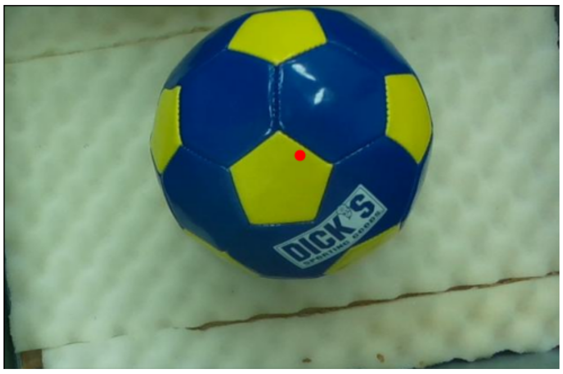

Figure 7.7: Example of a scene view with a ball, the red point indicates the grasp point predicted by the neural network.

In the Figure 7.7, we can observe an example of a failed grasp for the ball object, where grasps near the seams, even if close to the center of the object, are unsuccessful. These seams, being very small, are not detected by the camera, and thus, the information is not conveyed to the neural network, which perceives a smooth surface. Resolving errors of this nature would require adjustments to the suction cup and the material it is made of. However, addressing such issues goes beyond the scope of the thesis work.

In this example Figure 7.8, we can indeed observe a prediction point that is

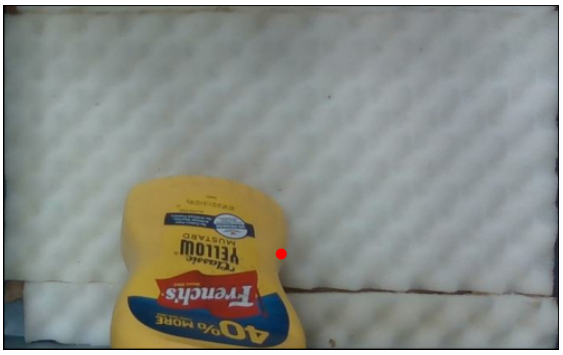

Figure 7.8: Example of a scene view with a detergent bottle, the red point indicates the grasp point predicted by the neural network.

inaccurately placed. The predicted point is on the edge of the detergent bottle, preventing the suction cup from adhering effectively. In this case, the detergent bottle is positioned obliquely on the edge of the container.

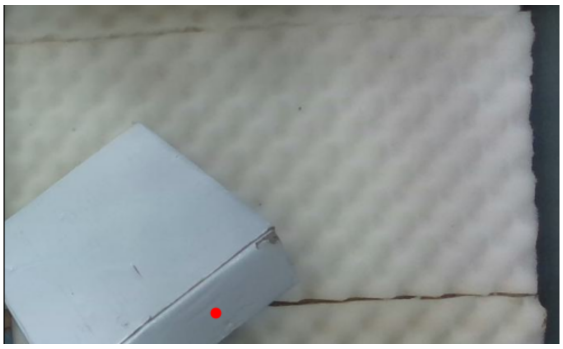

Figure 7.9: Example of a scene view with a box, the red point indicates the grasp point predicted by the neural network.

In this other example Figure 7.9, we can also observe a situation similar to the previous one where the box is placed on the edge of the container, resting on it at an oblique angle. However, in this case, the predicted grasp point is quite good as it is on the narrower edge of the box but still centrally located. The grasp failed because the robot approached that point with an incorrect orientation, likely caused by imprecise parameter choices for estimating the normal. This type of error has been found to be common for poses of this nature.

One of the two incorrectly predicted grasps for the bowl object by the neural network is illustrated in the Figure 7.10. As we can see in this case, it predicted a point on the edge of the bowl instead of on the flat surface.

These were some examples of failed grasps. As evident from the table, there are

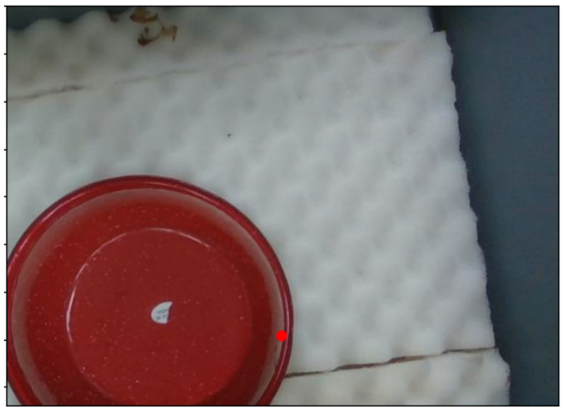

Figure 7.10: Example of a scene view with a bowl, the blue point indicates the grasp point predicted by the neural network.

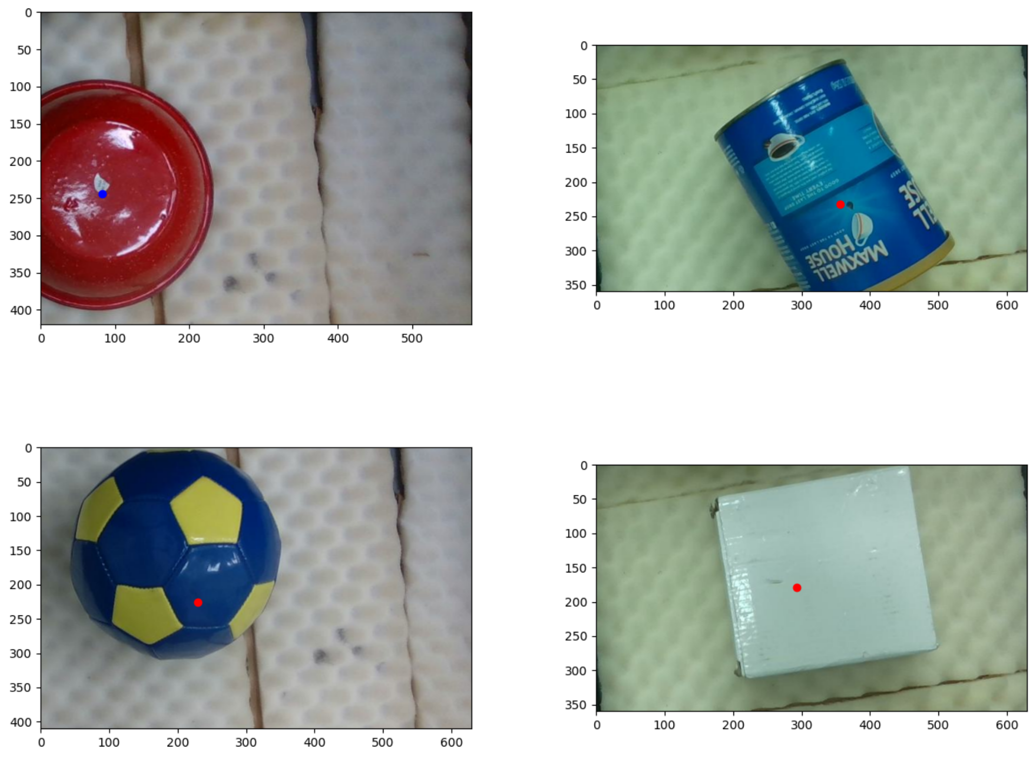

Figure 7.11: Examples of successful grasps for bowl, jar, ball and box

many other successful grasps, and in the Figure 7.11, we can see some of them.

Object	Success rate
Banana	70%
Screwdriver	75%
Ball	90%
Small box	85%
Capacitor	75%

Table 7.5: Table of success rate of the small object evaluated. 20 trials were carried out for each object

Small size objects

Regarding smaller or more challenging objects, we selected a banana, a screwdriver, a small ball, a box, and a capacitor. We chose these objects because they are thinner, such as the banana or the screwdriver, and being thinner, they have fewer good grasp points. Additionally, small errors are more likely to prevent the suction cup from adhering well to the surface. In the table, you can see the success rate for each object. The overall success rate is slightly lower, as expected, and is 79%. The most challenging objects to grasp are the banana, the screwdriver, and the capacitor. The capacitor, in particular, is very small, approximately 4 cm x 2 cm, so even a small error can lead to a failed grasp.

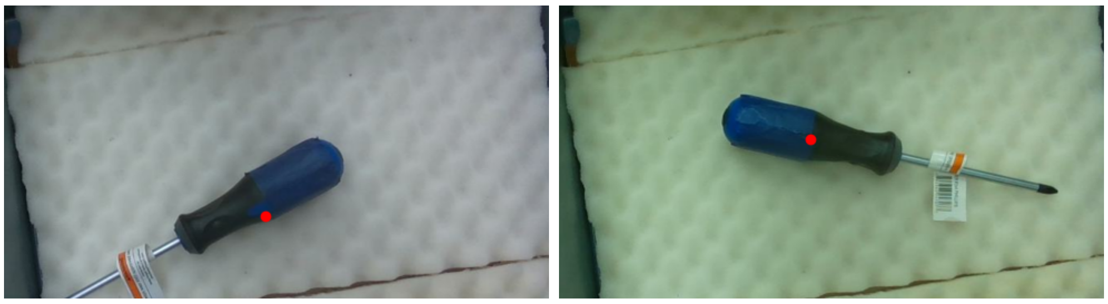

Figure 7.12: Two examples of neural network predictions for a scene containing a screwdriver. The prediction on the left failed, instead, the prediction on the right is succeeded.

In the Figure 7.12, 7.13, we observe the comparison between a correct grasp and an incorrect grasp for two objects belonging to this category.

Transparent object

As the last object, a transparent detergent bottle was tested to highlight the camera's difficulty in capturing transparent objects. In the Figure 7.14, only the

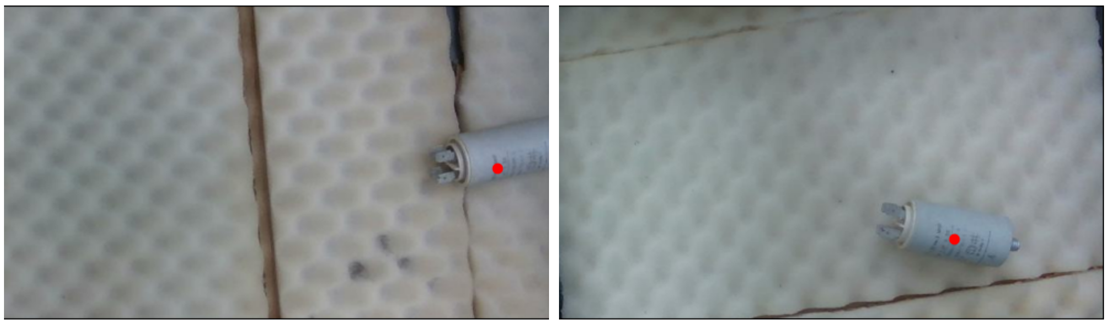

Figure 7.13: Two examples of neural network predictions for a scene containing a capacitor. The prediction on the left failed, instead, the prediction on the right is succeeded.

areas where the bottle's label is present contain information, and these are also the points where the network makes predictions. In this scenario, tests were conducted with only one object because, in the end, the network predicts grasp points primarily on the label rather than the transparent part, as the points on the transparent portion are sparse and noisy.

Object	Success rate
Transparent detergent	85%

Table 7.6: Table of success rate of the transparent object.

Conclusions

With these images, we have examined various scenarios leading to the failure of object grasping, which we can summarize as follows:

- Object-related causes: This category includes cases such as the ball, where the seams prevent the suction cup from creating a vacuum.
- Approach direction-related causes: Here, all failed grasps due to errors in estimating the normal of the grasp point are considered. For example, when the robot approaches with an incorrect orientation.
- Network-related causes: This category encompasses all grasps incorrectly predicted by the neural network, which are unrelated to the object or the normal estimation algorithm.

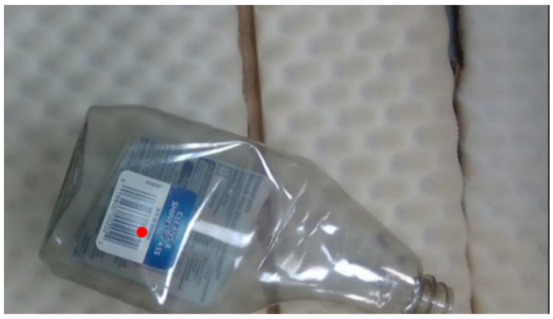

Figure 7.14: Example of a scene view with a transparent bottle, the red point indicates the grasp point predicted by the neural network.

Chapter 8

Conclusion

In conclusion, this research has delved into the intricate realm of robotic object manipulation, a pivotal aspect of robotics with applications spanning from industrial manufacturing to robot-assisted surgery. The focus has been on the generation of optimal grasp points, a fundamental component of successful object manipulation.

Two distinct approaches were explored for grasp point generation: classification and regression. The former offers a binary decision on the grasp quality of points, while the latter seeks to predict the coordinates of the optimal grasp point directly. The evaluation of these approaches was conducted using two different samplers, each with its unique methodology.

The first sampler employed an adaptive matrix selection method generated by a small neural network. This dynamic approach allows for adaptation to an object's shape, providing a flexible solution for optimal grasp point generation. The second sampler utilized the Chamfer distance metric to minimize the distance between sampled points and the points within the object's surface point cloud, enhancing precision in grasp point identification.

As we have observed, the proposed neural network for classification achieves very good results both in the case of classifying a single point and when classifying multiple points. In the former scenario, despite not surpassing the results obtained by GQCNN, a more traditional network that relies solely on information from the depth image for classification, our approach still produces results that, while somewhat inferior, are comparable. In the latter scenario, a direct comparison with other types of neural networks was not possible, as the approach we employed has not been tested by any other methods thus far. Even when evaluating the results qualitatively and conducting experiments with the real robot, we can affirm that the outcomes in this case are also highly satisfactory, achieving an overall accuracy of 89% and a success rate of 81%.

For the samplers the comparison we made shows that the new typology achieves remarkable results improving the metrics of unique points generated but having a coverage of the points slightly lower. At the same time, the sampler based on the selection matrix is lightweight, similar to the one based on chamfer distance, maintaining low network complexity to achieve the lowest possible prediction time. As we have seen, both samplers are not perfect and make errors, and there are certainly possibilities for further improvements to increase the success rate of both. However, the final result is satisfactory and establishes a foundation for future studies and enhancements.

8.1 Future works

All the trainings we conducted were carried out considering one object at a time, simplifying the task since a scene with multiple overlapping objects can increase complexity, involving interactions between objects and potentially reducing grasp points for each object. A future development could involve examining how this type of network performs on scenes with multiple objects, focusing on analyzing the sampler's ability to extract optimal points for all objects rather than concentrating on just one.

Another area for improvement concerns the number of trials conducted with the robot. The variety of different cases was limited, and more testing hours would be beneficial. This is especially crucial for industrial applications, where a robot needs to operate 24 hours a day, 7 days a week. During its operation, it will encounter a much larger variety of scenarios. In the trials conducted, efforts were made to test different object poses to cover as many scenarios as possible, but they are still limited. Additionally, the scene illumination seen by the robot remained nearly constant in all tests, and we are aware of how lighting conditions can affect the information obtained from the camera. Another aspect not evaluated is the use of different scenes; all scenes shared the same setup with a container containing the object. Therefore, the network's performance on other types of scenes could not be assessed.

Bibliography

[1] Jeffrey Mahler, Matthew Matl, Xinyu Liu, Albert Li, David Gealy, and Ken Goldberg. «Dex-Net 3.0: Computing Robust Vacuum Suction Grasp Targets in Point Clouds using a New Analytic Model and Deep Learning». In: (2017). DOI: `10.48550/arXiv.1709.06670` (cit. on pp. 2, 3, 22, 41).

[2] Hanwen Cao, Hao-Shu Fang, Wenhai Liu, and Cewu Lu. «SuctionNet-1Billion: A Large-Scale Benchmark for Suction Grasping». In: (2021). DOI: `10.48550/arXiv.2103.12311` (cit. on pp. 2, 3, 23).

[3] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. «Rethinking Atrous Convolution for Semantic Image Segmentation». In: (2017). DOI: `10.48550/arXiv.1706.05587` (cit. on p. 2).

[4] Hui Zhang, Yanming Wu, Eric Demeester, and Karel Kellens. «BIG-Net: Deep Learning for Grasping With a Bio-Inspired Soft Gripper». In: (2023). DOI: `10.1109/LRA.2022.3229237` (cit. on pp. 3, 4).

[5] Antonio Alliegro, Martin Rudorfer, Fabio Frattin, Aleš Leonardis, and Tatiana Tommasi. «End-to-End Learning to Grasp via Sampling from Object Point Clouds». In: (2022). DOI: `10.48550/arXiv.2203.05585` (cit. on pp. 4, 5, 36, 37).

[6] R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J. Guibas. «PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation». In: (2017), pp. 77–85. DOI: `10.1109/CVPR.2017.16` (cit. on pp. 11, 12).

[7] C. Qi, L. Yi, Hao Su, and Leonidas J. Guibas. «PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space». In: (2017). DOI: `10.48550/arXiv.1706.02413` (cit. on pp. 11–13).

[8] Oren Dovrat, Itai Lang, and Shai Avidan. «Learning to Sample». In: (2019). DOI: `10.48550/arXiv.1812.01659` (cit. on p. 15).

[9] Itai Lang, Asaf Manor, and Shai Avidan. «SampleNet: Differentiable Point Cloud Sampling». In: (2020). DOI: `10.48550/arXiv.1912.03663` (cit. on pp. 16, 17).

[10] Wang M., Chen Q., and Fu Z. «LSNet: Learned Sampling Network for 3D Object Detection from Point Clouds». In: (2022). DOI: 10.3390/rs14071539 (cit. on pp. 18–20).

[11] Sergey Ioffe and Christian Szegedy. «Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift». In: (2015). DOI: 10.48550/arXiv.1502.03167 (cit. on pp. 32, 35, 36).

[12] Abien Fred Agarap. «Deep Learning using Rectified Linear Units (ReLU)». In: (2015). DOI: 10.48550/arXiv.1803.08375 (cit. on pp. 32, 35, 36).

[13] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Abed Al Kader Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. «PointNeXt: Revisiting PointNet++ with Improved Training and Scaling Strategies». In: (2022). DOI: 10.48550/arXiv.2206.04670 (cit. on p. 33).