PointNet: Deep learning on Point Set for 3D Classification and Segmentation

Mohammad Fahes
mohammad.fahes@mines-paristech.fr

April 5, 2021

Abstract

This project is devoted to the study of the paper: "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation" [1]. The main idea is to develop an efficient neural network architecture to solve classification, part and semantic segmentation tasks for raw point cloud data.

1. Introduction

Point cloud data is a well known 3D data form, where there is no real connections between the points, and it is acquired by multiple used sensors, such as lidar and depth cameras. Suppose we have a scene, those sensors capture position information of the objects contained in it. Some interesting tasks in point cloud data are classification and segmentation. However, some acquisition-related and inherent problems, like sensors' noise and the sparsity of data per object, make these tasks challenging. Deep learning techniques proved to be adapted to this type of problems. A well known approach is to use CNNs on 3D grids corresponding to the point cloud [2]. However, this is inefficient: a cubic voxel grid of size n contains n^3 voxels. So the aim is to develop an efficient neural network for classification and segmentation tasks, which simply takes raw point clouds as input, and here pointNet comes into play [1].
To do so, the network architecture should take into consideration the three main constraints on point clouds:

1. Invariance to permutations of the input: point clouds being unordered, for n points there exist $n!$ permutations conserving the same object.
2. Invariance to rigid transformations, i.e. to every transformation that preserves the distance between every pair of points (e.g. rotations, translations).
3. Interaction among points, so that the network captures local structures from neighboring points. If this is not necessary for classification, where generally global features are used, it is for segmentation tasks where we must make use of local point features along with global ones.

2. Architecture

In this section, we present PointNet architecture, all along with an explanation about the manner this model verifies the three constraints presented in the introduction 1. The idea of PointNet is to approximate a function $f : 2^{\mathbb{R}^N} \to \mathbb{R}$ by a symmetric function $g : \underbrace{\mathbb{R}^K \times ... \times \mathbb{R}^K}_{n} \to \mathbb{R}$ applied on transformed points in a set of cardinality n, where the transformation function is $h : \mathbb{R}^N \to \mathbb{R}^K$. So the problem is formulated as: $f(\{x_1, ..., x_n\}) \approx g(h(x_1), ..., h(x_n))$. As it will be seen in the following, h is approximated by an MLPs-based network, and g by a composition of a one variable function and a max pooling.

2.1. MLPs

The pointNet architecture, as it is in the original paper, is shown in the figure 1. Two shared multi-layer perceptron (MLP) are used. Each of them is shared for each of the n points in the input. The first one is a two-layer MLP: the first layer maps each point from R^3 to $\mathbb{R}^{64}$, and the other one is a mapping from $\mathbb{R}^{64}$ to itself. This procedure is repeated in a three-layer perceptron where the first layer maps the n points from $\mathbb{R}^{64}$ to itself, the second one from $\mathbb{R}^{64}$ to $\mathbb{R}^{128}$, and the third one from $\mathbb{R}^{128}$ to $\mathbb{R}^{1024}$. So globally, the points are mapped from 3 dimensional space to a higher-dimensional embedding space (i.e. 1024 dimensions). At the output of the second MLP, a max-pooling layer serves to create a 1024-dimensional global feature vector. Finally, this vector is fed to a three-layer perceptron (A three-layer fully connected network) that maps it to k output classification scores.
We see in the architecture that for the segmentation task,

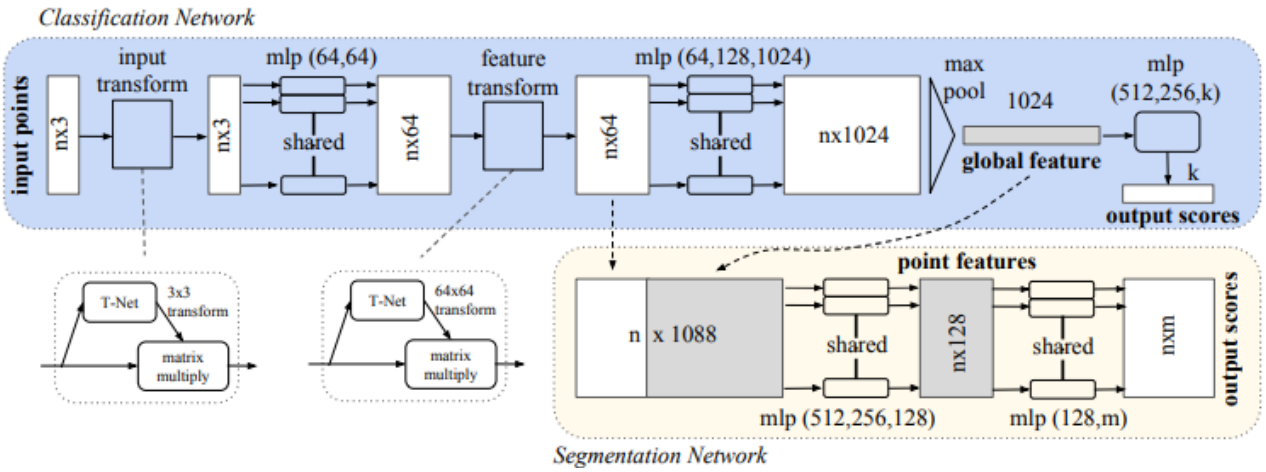

Figure 1: PointNet architecture

the network differs a little. Indeed, the output is this time $n \times m$, that is, each one of the n points is assigned to one of m segmentation classes. As invoked in the third constraint in the introduction 1, the segmentation makes use of both local and global point features, so the points in the 64-dimensional embedding space (input of the second mlp) which carry on local point features, are concatenated with the global feature vector. The result is then a $n \times 1088$ matrix. As in the classification network, two MLPs are used: a three-layer one which maps each of the n concatenated vectors from $\mathbb{R}^{1088}$ to $\mathbb{R}^{128}$, and a two-layer one mapping from $\mathbb{R}^{128}$ to $\mathbb{R}^m$. Here we attacked the part of the architecture only consisting of MLPs, which represents the core part in the classification and segmentation tasks.

2.2. Max pooling

From now on, we should attack the constraints on the point clouds, and how the architecture is specific to dealing with them. To respect the non-structure property, an intuitive way is to use symmetric functions, that is, functions that take n inputs and give the same output for all the $n!$ permutations. PointNet implements this required symmetry with max pooling.

As for the placement of the max pooling in the network, it seems convenient to use it just before getting the global feature vector, that is, feeding the fully-connected network with a vector which is permutation invariant.

We stress here that the authors of the article tested another symmetric functions, and max pooling lead to the better accuracy. We tested an alternative; the results are shown in 4.1.1.

2.3. T-nets

In 2.1, we stressed that the points interaction's constraint is bypassed by the concatenation of local point features along with global ones in the segmentation network. In 2.2, we justified that the max pooling layer used in the architecture aims to bypass the invariance to permutations constraint. What remains is to justify the usage of T-nets, and here we will see that the input transform and feature transform blocks are added to the network to verify the invariance to rigid transformations constraint.

A T-net is a regression network that predicts an affine tranformation matrix and mutiply it by the $n \times 3$ inputs. The architecture of this mini-network is inspired from the main architecture of PointNet. Indeed, MLPs are used as mapping to a higher dimensional space, the max-pooling is used to get the features' flattened vector in $\mathbb{R}^{1024}$(then it's mapped to $\mathbb{R}^{256}$ at the output of fully-connected layers), then globally trainable weighs and biases are combined to the 256-dimensional feature vector to get a 3×3 transformation matrix. The input transform block is shown in the figure 2.

The same approach is extended to the alignment of feature space (i.e. the 64-dimensional embedding space), and this is presented in the architecture by the "feature transform" block (see figure 1). The architecture of this T-net is the same as the first one, except that the trainable weighs and biaises are respectively 256×4096 matrix and 4096-dimensional vector. The result is a 64×64 transformation matrix. It is shown in the figure 3. To avoid overfitting, a regularization term is added to the softmax training loss and it forces the feature transformation matrix to approximate an orthogonal transformation which is supposed to not lose information in the input.

Finally we point out that, in the MLP global classification net, as in T-nets, all layers except the last one include ReLU and batch normalization.

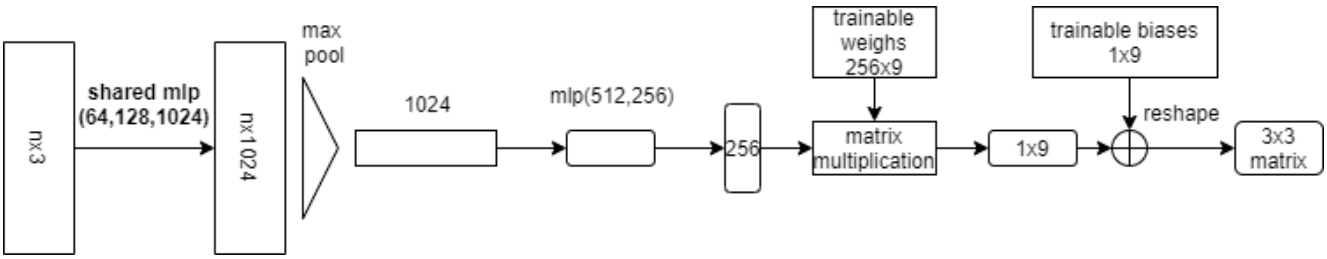

Figure 2: Architecture of 3x3 T-net

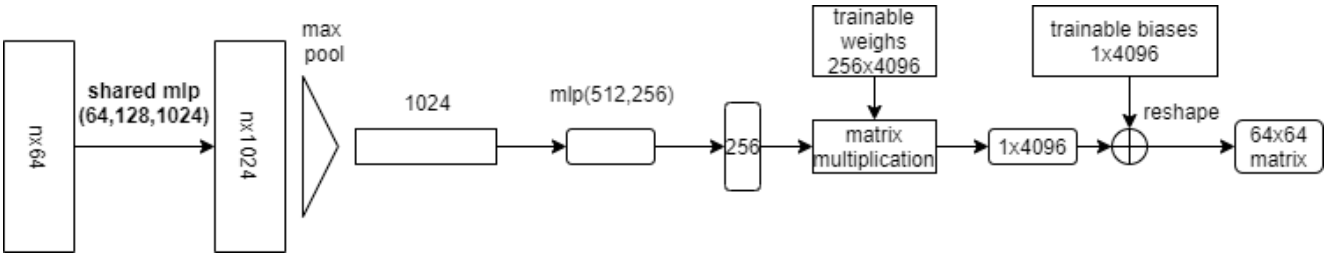

Figure 3: Architecture of 64x64 T-net

3. Theoretical analysis

First we study the network in the context of universal approximation. The aim is to establish universal approximation ability on the set of functions which are robust to small perturbations in the set of input points, and this latter will be formally the continuous set functions. Thus we want that $\forall \epsilon > 0, \exists \delta > 0, \forall S, S' \in \mathbb{X}\ , d_H(S, S') < \delta \implies |f(S) - f(S')| < \epsilon$, where $\mathbb{X} = \{S : S \subseteq [0,1]^m, |S| = n\}$, $f : \mathbb{X} \mapsto R$ and $d_H(.,.)$ designates the Hausdorff distance.
Starting from this definition of continuous set function, the first theorem of the paper states the existence of the two functions h and g that we talked about in the section 2. In section 4, we show the effect of the dimension of the max pooling layer (referred to as K in 2) on the classification accuracy.
The second theorem of the paper states that the function f (which will be the classification or segmentation score) remains the same up to a certain noise in the input points set, and that its value $f(S)$ is completely determined by a finite subset $C_S \subseteq S$, where $|C_S| \leq K$.

4. Experimental results

We tested PointNet with many variations, in order to see the impact of each part of the architecture and its importance to obtain high scores, in both classification and segmentation.

Datasets For the classification tasks, we used ModelNet40 dataset (the TP6's one), and shapeNet dataset for segmentation. We used google colab to make use of GPU.

4.1. Classification

For the classification task, we have tested the impact of dropping, adding, and modifying some parts of the paper's architecture. All the training in 4.1 was done on 30 epochs. ModelNet40 dataset that we used consists of 9843 samples for training and 2468 samples for testing.

4.1.1 Impact of symmetric function

We tested for the basic MLP PointNet (without input and feature transformation), the impact of the symmetric function used to get the global feature vector on the test accuracy. We got 84.44 % of accuracy for using "Maxpooling", and 80.43 % using "average pooling".

4.1.2 Impact of T-nets

First, we tried a basic version of PointNet without input and feature transformation. Next, we implemented it adding only the first T-net (input transform), then with input and feature transform (the full architecture). The results are shown in the table 1. We see that the adding of transformation blocks enhances the accuracy. However, using only

architecture	Test accuracy (%)
Basic version	83.9
Without feature transform	85.1
Full net	84

Table 1: Test accuracy for different architectures

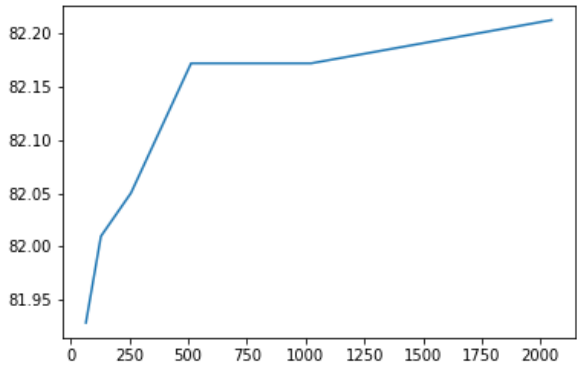

Figure 4: Influence of K on the accuracy of the model

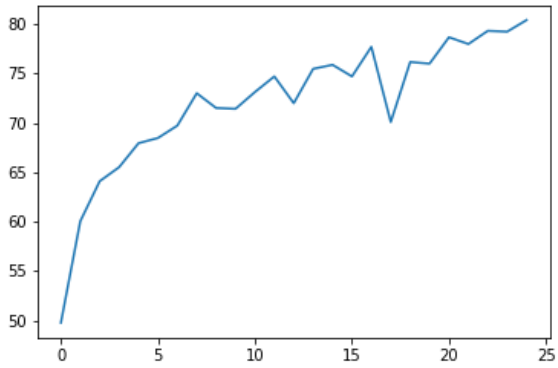

Figure 5: Accuracy score w.r.t epochs in basic MLP net (with average pooling used)

the input transforms yields to better results than using both input and feature transformations, and this normally should not happen. The cause should be the necessity of hyperparameter tuning, or training with more epochs, as the architecture is being more complex.

4.1.3 Impact of K

In this section we study the impact of the dimensionality K of the global feature vector on the accuracy of the model. We computed those accuracies for the full model (with input and feature transform) and the results are shown in the figure 4. As it can be seen, the classification accuracy increases with K, as the model complexifies more and more. In PointNet, K is chosen to be 1024. An interesting graph would be the study of the accuracy for different values of K and number of points in the input. We should expect that the accuracy will increase with the increasing of both variables (up to a certain limit in the value of K to avoid overfitting). Please note that in all the section above (4.1), the comparison is done on a fixed epoch number, and this could be misleading. As we can see in figure 5, we could do better in terms of accuracy score by increasing the number of epochs. Unfortunately, we were somehow obligated to fix the epochs' number at a limited value due to the training time consuming.

4.2. segmentation

For the segmentation, our tests were based on the github code (which we read and verified its sanity and its fitting to the paper), where we trained the model for 50 epochs. The part segmentation of the guitar, lamp, and chair are shown respectively in the figures 6, 7, and 8 as examples.

We point out that we computed the mIOU scores for several examples with and without feature transform. Some results are shown in the table 2.

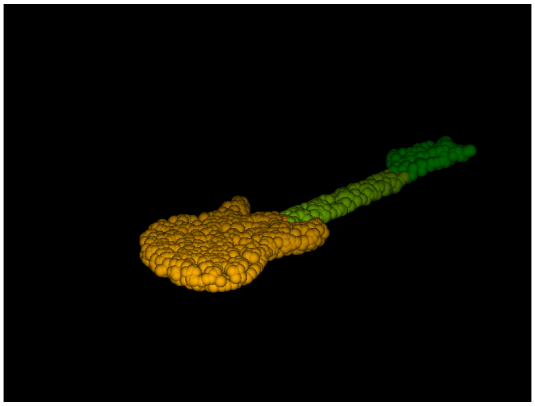

Figure 6: Guitar part segmentation

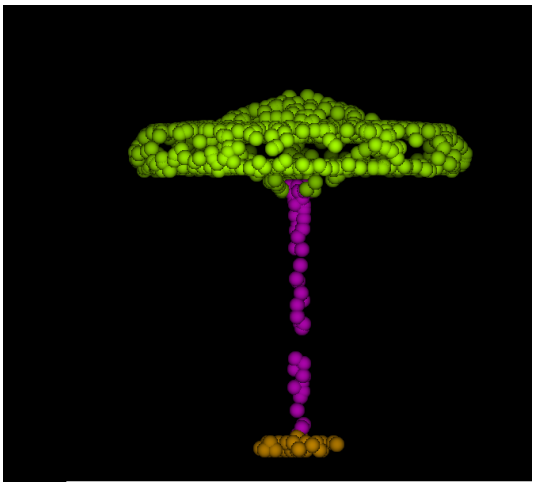

Figure 7: Lamp part segmentation

Class(mIOU)	feature transform	without feature transform
Guitar	0.849	0.852
Lamp	0.734	0.728
Chair	0.876	0.871
Motorbike	0.429	0.435

Table 2: Part segmentation results in terms of mIOU score

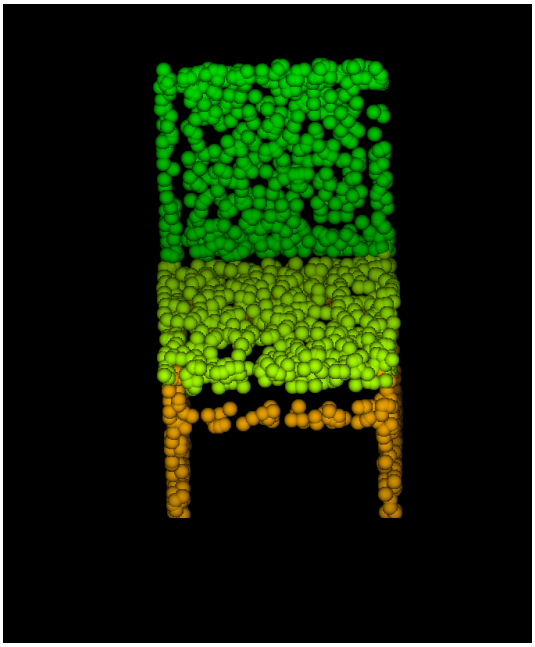

Figure 8: Chair part segmentation

5. Amelioration proposition

An idea of amelioration, which could let us gain in the classification and segmentation scores, would be the use of Topological data analysis (TDA) [3] and persistent homology [4] on point clouds as a prior information.
The key idea behind TDA is to extract shape-related features in a high-dimensional space. The main tool in TDA is persistent homology, which is an adaptation of homology to point clouds. As its name indicates, we want to know the topological features that are persistent at different spatial resolutions. So the intuition is to do so by tracking the topological features as a scale varies. This leads to separation between significant topological features, which could be used later to characterise the point cloud we are dealing with, and noise. Those topological features in persistent homology are Betti numbers, which characterise the number of holes in different dimensions.
For the readers who are not familiar with TDA, you can refer to the appendix A for the definitions of all the concepts used in persistent homology.

Thus the aim is to incorporate topological features during the training process and make use of the differentiable properties of persistent homology to construct a loss function including a topological term, in order to drive the training to give results topologically verified. To illustrate this, we take the example of classification of handwritten digits, where a network classifies an 8 digit as 0. Here comes the role of persistent homology, which will tell that in terms of β_1 (first Betti number), which corresponds to the persistent homology of dimension 1, and represents the number of independant cycles of dimension 1 which are not equivalent to the boundary of a 2-chain, β_1 of the shape of the digit 0 is equal to 1 whereas β_1 of the shape of the digit 8 is equal to 2. This comes as a penalization to the general loss function, and the classification would change its direction based on this penalization.
For the segmentation of point clouds, the same approach could be used. It demonstrated success and enhancement of performance scores on many datasets, as the article [5] shows (please note that the data used is not point clouds, however, the approach is the same).

Limitations As the proposed method seems to be efficient and applicable, a critical thinking on it lets us point out many limitations:

1. If the topology is correct but the segmentation map is not, the method has no effect.

2. If the prediction is far from the ground truth, it may not be possible to recover the correct one (2 connected components/ 1 connected component).

3. The persistent homology has high time and space complexity, which will increase significantly the time for segmentation to be done.

4. The prior knowledge of the topology is not trivial, even if it is known for an object, it depends on many other things (e.g. point of view, occlusion).

6. Conclusion

In this report, we showed the properties of PointNet and its adaptation (by design) to the constraints on point

clouds. We also tested many variations of the architecture and showed the influence of each modification on the scores in order to verify experimentally the role of each part of the architecture. Finally, we proposed a TDA method based on persistent homology to the improvement of the segmentation and classification scores.

References

[1] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 1

[2] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5648–5656, 2016. 1

[3] Herbert Edelsbrunner and John Harer. *Computational topology: an introduction*. American Mathematical Soc., 2010. 5

[4] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. In *Proceedings 41st annual symposium on foundations of computer science*, pages 454–463. IEEE, 2000. 5

[5] James Clough, Nicholas Byrne, Ilkay Oksuz, Veronika A Zimmer, Julia A Schnabel, and Andrew King. A topological loss function for deep-learning based image segmentation using persistent homology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 5

[6] Herbert Edelsbrunner and John Harer. Persistent homology-a survey. *Contemporary mathematics*, 453:257–282, 2008. 8

Appendix

A. Formal definitions

In this section we introduce some useful definitions to better understand persistent homology and the proposed approach. We recommend the reader who is not familiar with TDA to read these definitions in the same order we put them (the latter not being random). Each concept is a base to the understanding of the next ones, and all of them is necessary to understand the philosophy of persistent homology.

- **d-Simplex** σ : defined by its dimension d, it's the convex hull of d+1 points in $\mathbb{R}^n$ linearly independent ($0 \leq d \leq n$, where n is the dimension of the space in which the simplex is defined).
- **Face** $\tau \subseteq \sigma$: simplex defined from a subset of points defining σ.
- **Simplicial complex K**: It is a collection of simplices $\{\sigma_i\}$ verifying the two following conditions:
  1. $\sigma \in K,\ \tau \subseteq \sigma \Rightarrow \tau \in K$: All the faces of a simplex belonging to a simplicial complex K are in K.
  2. $(\sigma_i \cap \sigma_j) \subseteq \sigma_i, \sigma_j$: The intersection of two simplices in K is common face between them.
- **Vietoris-Rips complex**: It is the simplicial complex defined by all the simplices for which the pairwise distance between vertices is at most ϵ (for a fixed ϵ).
- **k-chains**: Consider a d-dimensional simplicial complex K. Let $k \in \{0, 1, ..., d\}$ and $\{\sigma_1, ..., \sigma_p\}$ the set of k-simplices of K. A k-chain is defined as follows: $c = \sum_{i=1}^{p} \epsilon_i \sigma_i$ where $\epsilon_i \in \mathbb{Z}/2\mathbb{Z} = \{0, 1\}$.
  $c + c' = \sum_{i=1}^{p} (\epsilon_i + \epsilon'_i)\sigma_i$ and $\lambda c = \sum_{i=1}^{p} (\lambda \epsilon_i)\sigma_i$.
  So we have a structure of a vector space over the field $\mathbb{Z}/2\mathbb{Z}$. The space of k-chains is then well defined.
- **Boundary operator** ∂: The boundary $\partial\sigma$ of a k-simplex σ is the sum of its (k-1)-faces. It is a (k-1)-chain. It is a linear map from the space of k-chains to the space of (k-1)-chains:

$$\begin{aligned} \partial_k : C_k(K) &\longrightarrow C_{k-1}(K) \\ c &\longrightarrow \partial_k(c) = \sum_{\sigma \in c} \partial_k \sigma \end{aligned}$$

- **Chain complex**: The chain complex associated to a d-dimensional complex K is constructed by taking the d-chain of K, and applying the boundary operator each time until we get 0.
  $C_d(K) \xrightarrow{\partial} C_{d-1}(K) \xrightarrow{\partial} ...C_{k+1}(K) \xrightarrow{\partial} C_k(K) \xrightarrow{\partial} ... \xrightarrow{\partial} C_1(K) \xrightarrow{\partial} C_0(K) \xrightarrow{\partial} 0$
- **k-cycles**: $Z_k(K) := Ker(\partial : C_k \to C_{k-1}) = \{c \in C_k : \partial c = \varnothing\}$
- **k-boundaries**: $B_k(K) := im(\partial : C_{k+1} \to C_k) = \{c \in C_k : \exists c' \in C_{k+1}, c = \partial c'\}$.
  We note that $B_k(K) \subset Z_k(K) \subset C_k(K)$.
- **Homology group**: The k-th homology group of K: $H_k(K) = \frac{Z_k}{B_k}$. We note that it's a vector space being the quotient of two vector spaces.
- **Betti number**: The k-th Betti number of K is: $\beta_k(K) = rank(H_k(K))$. It refers to the number of k-dimensional holes in a topological surface, that is, the k-dimensional cycles which are not the boundary of a (k+1)-dimensional object. β_0 is the number of connected components, β_1 is the number of circular holes, and β_2 is the number of 2-dimensional voids.
- **Filtration of simplicial complexes**: It is a nested sequence of subcomplexes such that:
  1. $\varnothing = K^0 \subset K^1 \subset ... \subset K^m = K$
  2. $K^{i+1} = K^i \cup \sigma^{i+1}$, where σ^{i+1} is a simplex of K.

A homology class α is born at K_i if it's not in the image of the map induced by the inclusion $K_{i-1} \subset K_i$. If the image of the map induced by $K_{i-1} \subset K_j$ contains the image of α and j is the least index for which this is true, we say that α dies entering K_j. The persistence of α is thus $j - i$ [6]. It is the "lifetime" of α. All the couples (birth-death), for a homology group, are represented in a persistence diagram, where the x-coordinate indicates the birth of a feature and the y-coordinate indicates its death. It is obvious that the points far from the diagonal are the most peristent, thus they represent the most important topological features.