POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Advanced Deep Learning Methods for Anomaly Detection in Point Clouds

Laurea Magistrale in Mathematical Engineering - Ingegneria Matematica

Author: Stefano Bruno Gusmeroli

Advisor: Prof. Giacomo Boracchi

Co-advisor: Luca Frittoli

Academic year: 2021-2022

1. Introduction

A point cloud is a compact and convenient way to represent a 3D object, aspect that has favoured the diffusion of such type of data over the last few years. Concomitantly, a literature of deep learning methods aimed to process them has flourished. Nonetheless, to our concern, we have noticed that anomaly detection has been much less regarded in such field of study. Motivated to explore this research area, in our work we present two novel methods to address unsupervised anomaly detection in point clouds.

2. Problem Formulation

We define a point cloud $\mathcal{P}$ as a pair $\mathcal{P} = (P, \varphi)$, where $P = \{x_i\}_i$ is an unordered set of points in a Euclidean space $\mathbb{R}^d$ and $\varphi : \mathbb{R}^d \to \mathbb{R}^I$ is a function that associates to each point $x \in \mathbb{R}^d$ some features $\varphi(x) \in \mathbb{R}^I$. Here we focus on 3D anomaly detection and we employ a point cloud to depict the surface of an object, hence $d = 3$. Our objective is to develop a method that, when it is presented with an element $\mathcal{P}$, is able to discriminate on whether it belongs to the normal class or not. The output of the method is an anomaly score $\mathcal{A}_S(\mathcal{P}) \in \mathbb{R}$ such that high values of it indicate that $\mathcal{P}$ is anomalous, whereas low values signify that $\mathcal{P}$ belongs to the normal class. Formally, the function $\mathcal{A}_S(\,\cdot\,)$ takes the name of *anomaly score function*.
We consider the framework of unsupervised anomaly detection, which means that the training samples are not labelled and that the model is trained using solely instances belonging to the normal class, as in [5], [2], [4], [3], [6].

3. Related Works

The points of the set P are unordered, from which it follows that every method that operates on point clouds is required to be invariant to the order of how the points are presented to it. To this end, PointNet uses an MLP to process every point and then aggregates all the extracted information by means of a symmetric function, such as a Max Pooling operator. ConvPoint introduces a point convolutional layer, which resembles the ones used in image processing. Similarly, [2] presents a novel point convolutional operation, used to implement the *composite layers*. A composite layer is made of a *spatial function* $s : \mathbb{R}^d \to \mathbb{R}^K$, deputed to extract information from the coordinates P, and of a *semantic function* f that combines the learnt information by s with the features φ. s is modelled by a Radial Basis Function Network of M centers $c_m \in \mathbb{R}^d$.

The convolutional operation is defined as:

$$\psi_j(y) = \sum_{x \in X_y} \sum_{i=1}^{I} \varphi_i(x) \sum_{k=1}^{K} w_{ijk} s_k(x - y)$$

with $\psi(y) \in \mathbb{R}^J$ being the new features of y. The convolutional window X_y is a set of points and it is defined by nearest neighbours.

Unsupervised anomaly detection. Reconstruction models [5] embed the input in a lower dimensional space and subsequently attempt to reconstruct it. The model is optimised to reconstruct uniquely normal instances and ideally shall fail to do so for anomalies. DeepSVDD aims to map all the normal instances inside a minimal enclosing hyper-sphere in a deep manner. Related to it, in DROCC [4] the elements are deemed to be anomalous if they lie outside the union of a collection of hyper-spheres centred at some typical training points. A different approach is based on applying a set of geometric transformations to the input and training a discriminator to predict which one had been used [3]. The discriminator, in order to distinguish between the transformed versions, is required to capture salient geometrical features, some of which are characteristic of the normal class. The drawback of this method is having to manually define the transformations a priori. In this line of work, NeuTraL [6] circumvents such limitation by learning directly the transformations by means of neural networks and by introducing the novel *Deterministic Contrastive Loss*.

4. Proposed Solutions

We propose two different methods to address the anomaly detection task on point clouds. The first one is an extension of the Deep Robust One Class Classification approach that is originally applied to images and tabular data in [4]. The second takes inspiration from [6], as it proved to be effective on tabular data and time series.

4.1. DROCC for point clouds

The underlying hypothesis of DROCC is that *"the set of typical normal elements lies on a low dimensional locally linear manifold that is well sampled"* [4]. We retain this assumption to be true also in the case of point clouds and, to guarantee the quality of the sampling, we consider normal classes made of thousands of elements.
The learning process relies on the discrimination between normal instances $\mathcal{P}$ and synthetically generated anomalies $\tilde{\mathcal{P}}$. To this end, we have adapted the original loss to the case of point clouds. Specifically, given a set of point clouds $\{\mathcal{P}_m\}_{m=1}^{M}$, the loss function is defined as:

$$\begin{aligned}\mathcal{L}_{\text{DROCC}} = \lambda \| \theta \|^2 + \sum_{m=1}^{M} \big[\ell(\phi_\theta(\mathcal{P}_m), 0) + \\ + \mu \max_{\tilde{\mathcal{P}}_m \in N_m(r)} \ell(\phi_\theta(\tilde{\mathcal{P}}_m), 1) \big]\end{aligned} \tag{1}$$

where $\| \theta \|^2$ is the L^2-regularisation term, λ, μ are hyper-parameters and $\ell(\cdot)$ is the binary cross-entropy loss. The function ϕ_θ processes a point cloud $\mathcal{P}$ and returns its anomaly score $\phi_\theta(\mathcal{P})$. The normal class is identified by the label "0", whereas anomalies by "1", therefore the aim of ℓ is to classify the normal instances $\mathcal{P}$ as belonging to the normal class and the adversarially generated examples $\tilde{\mathcal{P}}$ as anomalies.

4.1.1 Anomalies' generation

One of the pillars of DROCC is the generation of synthetic anomalous examples $\tilde{\mathcal{P}}$ by a gradient ascent phase during training. The anomaly generation problem can be defined as:

$$\max_{\tilde{\mathcal{P}} \in N(r)} \ell\left(\phi_\theta(\tilde{\mathcal{P}}), 1\right) \tag{2}$$

which translates into maximising the agreement between the prediction $\phi_\theta(\tilde{\mathcal{P}})$ of the network and the label "1" according to the function ℓ.
Since a point cloud $\mathcal{P} = (P, \varphi)$ is made of two components, $\mathcal{P}$ might be deemed to be anomalous based on the coordinates P, on the features φ or on both. However, since both the datasets that we use in our experiments contain constant features ($\varphi \equiv 1$), we perform the adversarial search of anomalies solely in the space of coordinates. Thus, given P, the most adversarial element $\tilde{P}$ is sought after within the set:

$$N(r) := \left\{ r \leq \| \tilde{P} - P \|_2 \leq \gamma \cdot r \right\} \tag{3}$$

where r is the *radius* and γ is an upper bound which helps in stabilising the training process. $N(r)$ contains elements at least at distance r from P, which are therefore regarded as anomalous in accordance with the *manifold hypothesis*. Given a set of coordinates P, we first add to each component a zero-mean Gaussian noise of standard deviation σ and we indicate the resulting set by $P + h_0$. After that, we optimise (2) for $s = 1, ..., S$ iterations. To do so we compute the gradient of $\ell(\phi_\theta((P + h_{s-1}, \varphi))$ w.r.t. h_{s-1} and

we make a step of length η in that direction:

$$h_s = h_{s-1} + \eta \frac{\nabla_{h,\,s-1}\ell(h_{s-1})}{\| \nabla_{h,\,s-1}\ell(h_{s-1}) \|} \tag{4}$$

Following [4], we project h_s (4) into the set $N(r)$ (3) by computing the scalar multiplication:

$$h_s = \alpha \cdot h_s \qquad \text{where}$$

$$\alpha = \begin{cases} \gamma \cdot r / \| h_s \| & if \ \| h_s \| \geq \gamma \cdot r \\ r / \| h_s \| & if \ \| h_s \| \leq r \\ 1 & otherwise \end{cases} \tag{5}$$

Ultimately, we set $\tilde{\mathcal{P}} := (P + h_S, \varphi)$. The outline of the DROCC algorithm is presented in Alg.1.

Algorithm 1 DROCC for Point Cloud Anomaly Detection *(One epoch)*

Input: $\{\mathcal{P}_j\}_j$

for each batch $\mathcal{B}_b$, $b = 1, ..., B$ **do**
  for each point cloud $\mathcal{P}_m \in \mathcal{B}_b$ **do**
    Sample $h_0 \sim \mathcal{N}(0, \sigma^2 \cdot I)$
    for each step $s = 1, ..., S$ **do**
      $\ell(h_{s-1}) := \ell\big[\phi_\theta(P + h_{s-1}, \varphi), 1\big]$
      $h_s = h_{s-1} + \eta \frac{\nabla_{h,\,s-1}\ell(h_{s-1})}{\| \nabla_{h,\,s-1}\ell(h_{s-1}) \|}$
      $h_s = \alpha \cdot h_s$ (5)
    end for
    $\tilde{\mathcal{P}}_m := (P + h_S, \varphi)$
  end for
  $\mathcal{L} = \sum_{\mathcal{P}_m \in \mathcal{B}_b} \big[\ell(\phi_\theta(\mathcal{P}_m), 0) + \mu \max_{\tilde{\mathcal{P}}_m \in N_m(r)} \ell(\phi_\theta(\tilde{\mathcal{P}}_m), 1)\big] + \lambda \| \theta \|^2$
  $\theta = \theta - Gradient\_step\{\mathcal{L}\}$
end for

Table 1: Architecture of **ADCompositeNet3**; J is the number of output features, $|X_y|$ the cardinality of neighbourhoods and $|Q|$ of the output points. BN stands for Batch Normalisation

Layer type	J	$\lvert X_y \rvert$	$\lvert Q \rvert$
Composite+BN+ReLU	16	32	256
Composite+BN+ReLU	48	16	64
Composite+BN+ReLU	96	16	1
Fully Connected	1	-	-

4.1.2 Architecture of ϕ_θ

We have designed various architectures primarily based on the composite layers [2], with an eye to investigating their effectiveness in this context. An example is reported in Tab.1. In addition, we have also tested other types of networks such as a ConvPoint and a PointNet.

4.2. NeuTraL for point cloud AD

The second method that we have developed takes inspiration from [6] and it is made of three components: a pre-trained feature extractor, a set of learnable transformations and an encoder. The **feature extractor** ϕ_{PC} processes a point cloud $\mathcal{P}$ and it returns a global vector descriptor $\phi_{PC}(\mathcal{P}) \in \mathbb{R}^{\bar{J}}$. For this role, we have employed a CompositeNet [2] made of five convolutional layers, whose architecture is in Tab.2.

Table 2: **Feature extractor** architecture

Layer type	J	$\lvert X_y \rvert$	$\lvert Q \rvert$
Composite+BN+ReLU	64	32	1024
Composite+BN+ReLU	128	32	256
Composite+BN+ReLU	256	24	64
Composite+BN+ReLU	512	16	16
Composite	1024	16	1

The **set of learnable transformations** $\mathcal{T} = \{T_1, ..., T_K\}$ is constituted by K functions $T_k : \mathbb{R}^{\bar{J}} \to \mathbb{R}^{\bar{J}}$, whose parameters are denoted by ϑ_k. Each transformation takes as input a global descriptor vector $\phi_{PC}(\mathcal{P})$ and it returns a vector in $\mathbb{R}^{\bar{J}}$ of the same dimension. To model each transformation we have used an MLP made of 3 fully connected layers of input and output dimension equal to 1024. A ReLU activation function is employed after every layer but the last one.
Ultimately, the **encoder** ϕ_{enc} maps the transformed versions of a same element into a lower dimensional space where ideally they are more easily distinguishable [6]. The encoder is therefore a function $\phi_{enc} : \mathbb{R}^{\bar{J}} \to \mathbb{R}^{L}$. The same encoder is used to map all the transformed versions and we have modelled it with a simple MLP:

$$FC1(1024, 640) \to ReLU \to FC2(640, 256)$$

The key aspect of this method is the loss function that is minimised, namely the Deterministic Contrastive Loss **(DCL)** [6]. Let us first define $p_k = T_k(\phi_{PC}(\mathcal{P}))$, $p_l = T_l(\phi_{PC}(\mathcal{P}))$ and:

$$h(\mathcal{P}_k, \mathcal{P}_l) := \exp\big\{ sim\big[\phi_{enc}(p_k), \phi_{enc}(p_l)\big]/\tau \big\}$$

where τ is a temperature hyper-parameter and $sim[\cdot, \cdot]$ is the cosine similarity function. The **DCL** function can be therefore written as:

$$\mathcal{L} := \mathbb{E}_{\mathcal{P} \sim \mathcal{D}} \left[-\sum_{k=1}^{K} \log \frac{h(\mathcal{P}_k, \mathcal{P})}{h(\mathcal{P}_k, \mathcal{P}) + \sum_{l \neq k} h(\mathcal{P}_k, \mathcal{P}_l)} \right]$$

When $\mathcal{L}$ is minimised, the numerator pushes the embedding of each transformed version $T_k(\phi_{PC}(\mathcal{P}))$ close to the one of the original element $\phi_{PC}(\mathcal{P})$, fact that incentives the transformations to retain relevant semantic information. On the other side, the denominator encourages the transformed versions to be dissimilar from each other by pulling away the relative embed-

dings. At evaluation time, as anomaly score it is employed:

$$\mathcal{A}_S(\mathcal{P}) = -\sum_{k=1}^{K} \log \frac{h(\mathcal{P}_k, \mathcal{P})}{h(\mathcal{P}_k, \mathcal{P}) + \sum_{l \neq k} h(\mathcal{P}_k, \mathcal{P}_l)}$$

5. Experiments

In order to assess the anomaly detection capabilities of our methods, we have conducted several experiments on two datasets: ShapeNet7C and ModelNet40. The setup that we have adopted is the *1 Vs. All*, which means that we regard one class at the time as "normal" and all the others as anomalies. Each time we train our model solely on the normal class and this process is repeated for each class of the dataset. ShapeNet7C is a subset made of the 7 most numerous classes of ShapeNet; it contains 24417 training and 6100 test samples. ModelNet40 is made of 40 classes of every-day object, for a total of 12311 instances, 9843 of which are used for training. The objects of both datasets are synthetically generated, hence they are not occluded and they have a uniform density. In our experiments we always use 1024 points and, for NeuTraL, $K = 15$ learnable transformations.
We employ the AUC as evaluation metric, which measures how well a model is able to separate the data into two classes. The AUC ranges from 0 to 1 and a random guesser makes register an AUC of 0.5. Thus, values below 0.5 indicate that the model performs worse than one without any knowledge of the data.
To the best of our knowledge, the only works that address our problem in a similar way to how we intend to, are [5], [1] and [2]. **VAE** [5] proposes a Variational Autoencoder trained to compress and subsequently reconstruct point clouds. **IFOR** [2] is an isolation forest on the handcrafted Global Orthographic Object Descriptors. **DeepSVDD** [1] is an extension of the original DeepSVDD. **Self-Sup** [2] is an application to point clouds of [3]. In this case, 8 rotations along a horizontal axis have been employed as set of transformations.

5.1. DROCC results

To test the effectiveness of the composite layers [2] we have constructed different architectures based on them. In addition to the convolutional ADCompositeNet3 (Conv3) in Tab.1, we have created a network with the same architecture using the aggregate layers (Aggr3). Furthermore, we have analysed an *ADCompositeNet5* (Conv5) network that is composed of five convolutional layers and is similar to [2]. Finally, we have considered *ADCompositeNet3b* (Conv3b), a slight variation of ADCompositeNet3 that uses two fully connected layers instead of one, with the first having output dimension 32.

Table 3: DROCC: Results (AUC) for different architectures based on the composite layers

Class		Conv5	Conv3	Conv3b	Aggr3
0	Airplane	0,7925	0,7909	**0,8182**	0,7709
1	Car	**0,6936**	0,6370	0,6479	0,6311
2	Chair	0,5871	**0,7336**	0,7315	0,7266
3	Lamp	0,6426	**0,6821**	0,5942	0,5895
4	Table	0,5864	0,7816	0,7524	**0,8198**
5	Sofa	**0,7564**	0,6935	0,6999	0,6420
6	Rifle	0,8537	0,8515	0,8847	**0,9015**
Average AUC		0,7018	**0,7386**	0,7327	0,7259
Average rank		2,57	**2,29**	**2,29**	2,86

As we can see from Tab.3, there is not a clearly predominant architecture. Nevertheless, we can notice that ADCompositeNet5 tends to perform worse than the others, fact that can be explained by its more complex architecture. Moreover it seems that the convolutional composite layers yield slightly better results than their aggregate counterpart, probably thanks to their simpler structure. On this basis, we speculate that our DROCC method reaches better results when simple architectures are used in the role of ϕ_θ.

Table 4: DROCC: Results in terms of AUC when different networks are employed

Class		ConvPt3	PointNet	Composite3
0	Airplane	0,6738	**0,8067**	0,7909
1	Car	0,5218	**0,7635**	0,6370
2	Chair	0,6374	0,6024	**0,7336**
3	Lamp	0,4829	0,6734	**0,6821**
4	Table	0,6973	0,6522	**0,7816**
5	Sofa	0,5465	0,6711	**0,6935**
6	Rifle	0,7440	**0,8612**	0,8515
Average AUC		0,6148	0,7186	**0,7386**
Average rank		2,71	1,86	**1,43**

A second group of experiments explores the performance of our DROCC method when different types of networks are adopted. Besides ADCompositeNet3 (Tab.1), we consider a ConvPoint network of same architecture, choice that is done to make the confront as fair as possible. Nevertheless, due to their different structures, *ConvPoint3* counts 457601 learnable parameters, whereas ADCompositeNet3 just 140423. For the comparison, we consider also a more tra-

Table 5: Results in terms of AUC of point cloud anomaly detection methods on ShapeNet7C

Class		IFOR [2]	VAE [5]	DeepSVDD [1]	DROCC (Ours)	Self-Sup. [2]	NeuTraL (Ours)
0	Airplane	0,912	0,747	0,6898	0,7909	0,9700	**0,9988**
1	Car	0,712	0,757	0,6217	0,6370	0,9720	**0,9984**
2	Chair	0,571	0,931	0,6758	0,7336	0,9410	**0,9573**
3	Lamp	0,962	0,907	0,6410	0,6821	0,4210	**0,9627**
4	Table	0,883	0,839	0,6585	0,7816	0,8540	**0,9916**
5	Sofa	0,986	0,777	0,5834	0,6935	0,9440	**0,9882**
6	Rifle	0,475	0,382	0,7422	0,8515	0,9770	**0,9984**
Average AUC		0,7859	0,7629	0,6589	0,7386	0,8684	**0,9851**
Average rank		3,43	4,00	5,43	4,29	2,86	**1,00**

ditional PointNet, which, in spite of the rather simple architecture, has 736641 parameters.
From Tab.4 we can see that, despite the similar structure, on average ADCompositeNet3 outperforms ConvPoint3 of around 10 % of AUC. As we have alluded to before, this is probably due to the higher complexity of the latter. However, the results of PointNet are on par with the ones of ADCompositeNet3, in spite of the highest number of parameters.
Collecting the above observations, we advocate that the optimisation process of the method we developed can be quite insidious, hence we recommend employing networks with a simple design in the role of $\phi_\theta(\cdot)$ and eventually prefer a low number of parameters. Sophisticated architectures that count millions of parameters tend to be not well suited to our DROCC method for anomaly detection on point clouds.
Ultimately, from the confront in Tab.5, we can see that our DROCC method performs better than the DeepSVDD from [1], the most similar method among the ones considered. Moreover, it is almost on par with VAE [5], which however uses 2048 points. Generally speaking, DROCC for point clouds does not prove to be extremely effective overall, as it is outperformed on several classes by the baseline IFOR [2]. Nevertheless, among these four methods, it does not emerge that one consistently performs better than the others on every class, as they all have their shortcomings. Indeed, apart from DeepSVDD and DROCC, all the others make register values of AUC below the 0.5 threshold on certain classes, hence performing worse than a random guesser which has no knowledge of the data.

5.2. NeuTraL on ShapeNet7C

The NeuTraL model that we employ for point cloud anomaly detection on ShapeNet7C makes use of a feature extractor pre-trained on ModelNet40. At this regard, we have adapted the architecture in Tab.2 by adding a final fully connected layer and we have optimised the resulting network to perform the classification task on a subset of ModelNet40. More precisely, we have removed 8 classes of objects similar to the ones of ShapeNet7C in order not to expose the ϕ_{PC} to such categories during pre-training. The resulting dataset *ModelNet32* counts 6645 samples divided into 32 classes. After pre-training, we remove the added classification head and we employ ϕ_{PC} to process the input point clouds. We have trained each model five times in order to make the results less subject to variability and in Tab.5 we report the mean AUC obtained. As we can see, our Neural Transformation Learning method for point cloud anomaly detection achieves impressive performance. In fact, on four out of the seven classes (namely "airplane", "car", "table" and "rifle") of ShapeNet7C, it reaches values of AUC greater than 0.99. From this comparison, it emerges that our NeuTraL method is the best performing one on the ShapeNet7C dataset, by making register the highest AUC on every class of it. Our model outperforms the shallow baseline IFOR [2], VAE [5], DeepSVDD [1] and DROCC by a very large margin and, in so doing, it sets the new state of the art. More precisely, it achieves an average AUC of 0.9851, which improves the previous best result of more than 10 % and it approaches the perfect value of 1.

5.3. NeuTraL with few samples

To further explore the performance of our NeuTraL method, we have inverted the roles of the aforementioned datasets by pre-training ϕ_{PC} on ShapeNet7C and performing anomaly detection on ModelNet32. We believe that this scenario is much more challenging for two reasons. The first is because ϕ_{PC} is pre-trained on a dataset containing fewer classes, hence it is exposed to a more limited selection of objects and conse-

quently of features. The second motivation is that the anomaly detection model is trained on much smaller normal classes, with some of them counting less than ninety samples. We train each model five times as previously in Sec.5.2.

Table 6: NeuTraL on ModelNet32. AUC averaged over 5 runs and Std. dev refers to their standard deviation. We report the number of training samples and the original label of each class in ModelNet40

Class		AUC	Std. dev.	#Train
1	bathtub	**0,9672**	0,004525	106
2	bed	**0,9939**	0,000570	515
3	bench	**0,9259**	0,003129	173
4	bookshelf	**0,9817**	0,001611	572
5	bottle	**0,9949**	0,000367	335
6	bowl	**0,9753**	0,003333	64
9	cone	**0,9787**	0,000870	167
10	cup	**0,9255**	0,000609	79
11	curtain	**0,9788**	0,001358	138
13	door	**0,9659**	0,003784	109
14	dresser	**0,9680**	0,001279	200
15	flower pot	**0,8536**	0,007559	149
16	glass box	**0,9690**	0,000302	171
17	guitar	**0,9940**	0,001005	155
18	keyboard	**0,9990**	0,000512	145
20	laptop	**0,9997**	0,000068	149
21	mantel	**0,9481**	0,003124	284
22	monitor	**0,9951**	0,000453	465
23	night stand	**0,9503**	0,000966	200
24	person	**0,9704**	0,001575	88
25	piano	**0,9073**	0,000879	231
26	plant	**0,9556**	0,002248	240
27	radio	**0,7835**	0,006306	104
28	range hood	**0,9641**	0,004160	115
29	sink	**0,8296**	0,008381	128
31	stairs	**0,8106**	0,020754	124
34	tent	**0,9713**	0,001122	163
35	toilet	**0,9947**	0,000683	344
36	tv stand	**0,9575**	0,000800	267
37	vase	**0,9397**	0,001043	475
38	wardrobe	**0,9188**	0,009396	87
39	xbox	**0,8923**	0,008521	103
Average AUC		**0,9456**	0,001429	6645

As we can see from Tab.6, our NeuTraL method for point cloud anomaly detection achieves excellent results also in this scenario, as the mean AUC over the 32 classes is of 0.9456. On top of it, our model reaches values of AUC above 0.99 on several classes. More remarkably, on "keyboard" it makes register an AUC of 0.999 and on "laptop" of even 0.9997, values astoundingly close to the perfect score of 1. Notably, these two classes count less than 150 training samples. The lowest value of AUC is recorded on "radio", on which it obtains a value of AUC of "only" 0.7835. We speculate that such behaviour is related to the low cardinality of said class and with the high variability within the class.

We hypothesise that these outstanding results are also thanks to the pre-trained feature extractor $\phi_{PC}(\cdot)$. This said, the main criticism may concern the additional data required. At this regard, our model proved to excel in both the considered scenarios, albeit being very different. On this ground, we speculate that the amount of pre-training data and the number of different categories used are not so fundamental to the good success of our anomaly detection method, as long as the extracted features generalise well. On the other side, the pre-training procedure demonstrated to be a very effective strategy in the context of anomaly detection.

6. Conclusion

In this work we have introduced two methods that address anomaly detection in point clouds. Our DROCC method has proved to be on par with some similar ones. Besides, our NeuTraL model, by leveraging a pre-trained feature extractor, has demonstrated impressive performance. It has outperformed every other method considered, making register values of AUC astoundingly close to 1 and hence approaching the perfect classifier.

References

[1] A. Floris. Composite convolution for 3d point clouds, 2021. M.Sc.Thesis, Politecnico di Milano.

[2] A. Floris, L. Frittoli, D. Carrera, and G. Boracchi. Composite layers for deep anomaly detection on 3d point clouds. *arXiv:2209.11796*, 2022.

[3] I. Golan and R. El-Yaniv. Deep anomaly detection using geometric transformations. *NeurIPS*, 31, 2018.

[4] S. Goyal, A. Raghunathan, M. Jain, H. V. Simhadri, and P. Jain. Drocc: Deep robust one-class classification. In *International Conference on Machine Learning (ICML)*, pages 3711–3721. PMLR, 2020.

[5] M. Masuda, R. Hachiuma, R. Fujii, H. Saito, and Y. Sekikawa. Toward unsupervised 3d point cloud anomaly detection using variational autoencoder. In *International Conference on Image Processing*, pages 3118–3122. IEEE, 2021.

[6] C. Qiu, T. Pfrommer, M. Kloft, S. Mandt, and M. Rudolph. Neural transformation learning for deep anomaly detection beyond images. In *ICML*, pages 8703–8714. PMLR, 2021.