16th International Conference on Machine Vision Applications (MVA)
National Olympics Memorial Youth Center, Tokyo, Japan, May 27-31, 2019.

# Zero-shot Learning of 3D Point Cloud Objects

Ali Cheraghian, Shafin Rahman and Lars Petersson
Australian National University, Data61-CSIRO
firstname.lastname@anu.edu.au

## Abstract

*Recent deep learning architectures can recognize instances of 3D point cloud objects of previously seen classes quite well. At the same time, current 3D depth camera technology allows generating/segmenting a large amount of 3D point cloud objects from an arbitrary scene, for which there is no previously seen training data. A challenge for a 3D point cloud recognition system is, then, to classify objects from new, unseen, classes. This issue can be resolved by adopting a zero-shot learning (ZSL) approach for 3D data, similar to the 2D image version of the same problem. ZSL attempts to classify unseen objects by comparing semantic information (attribute/word vector) of seen and unseen classes. Here, we adapt several recent 3D point cloud recognition systems to the ZSL setting with some changes to their architectures. To the best of our knowledge, this is the first attempt to classify unseen 3D point cloud objects in the ZSL setting. A standard protocol (which includes the choice of datasets and the seen/unseen split) to evaluate such systems is also proposed. Baseline performances are reported using the new protocol on the investigated models. This investigation throws a new challenge to the 3D point cloud recognition community that may instigate numerous future works.*

## 1 Introduction

In recent years, a number of methods have been introduced addressing the problem of 3D object classification of point clouds [18, 28, 10]. They have achieved outstanding accuracy on available 3D datasets, in most cases reaching a greater performance than 90%. This achievement is due to employing deep end-to-end learning on point cloud objects/scenes. Classical attempts of this kind requires numerous pre-processing steps involving voxel representations of 3D models [29], projecting it to 2D spaces [23], using pre-trained networks like VGG [22] etc. Those approaches are not only dependent on higher-end hardware but also not extendable to scene understanding/point classification/shape completion. The emergence of end-to-end learning on 3D point cloud data has solved most of the previous problems using one single deep neural network.

Nowadays, thanks to the availability of 3D depth cameras, obtaining 3D models of objects and environments are easier than before [2, 6]. This has opened up the space of potential applications, and also brings with it new challenges. For example, it is a likely scenario that we have access to, or is capturing, 3D models for which we do not have any labels. Making use of such data is a challenge, and we are in this paper exploring whether a zero-shot learning (ZSL) approach can be utilized (See Figure 1). To the best of our knowledge, this is the first time ZSL has been applied to 3D object recognition. In the 2D image domain, on the contrary, there are many works addressing the ZSL problem in the past few years [19, 32, 9, 3]. A general ZSL architecture introduces semantic information (like attributes [8] or word vectors [14, 16]) of classes to transfer the knowledge from the seen classes to the unseen classes. During training of ZSL, many methods convert image features to the semantic embedding [8] or a latent space [30], or vice versa [32]. Later, in testing, instances of unseen classes are projected to the same space learned during training, to predict a matching score based on the similarity between the projected embedding and the unseen semantic embedding. With the motivation from the 2D version of the ZSL problem, we use semantic information from classes inside the deep network to classify unseen 3D point cloud objects.

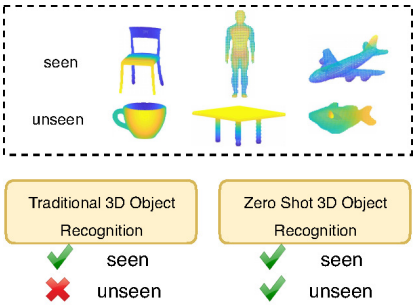


Figure 1. Traditional 3D point cloud recognition systems can only classify objects from seen classes. However, adopting a ZSL approach can enable a system to classify objects from classes that are not observed during training.

In this paper, we conduct a series of experiments utilising two popular structures traditionally used for feature extraction in 3D point clouds. These are PointNet [17] and EdgeConv [28]. We combine these with two pooling methods, Maxpooling and NetVlad [1]. With the help of these base architectures, we build a new structure that combines point cloud features with word vector semantic features thereby enabling the classification of previously unseen 3D classes. The-

ory is here provided as to how to combine semantic word vectors and adapt the previous structure to perform the ZSL task. Moreover, based on our experiments, we demonstrate that our proposed framework for classification of unseen 3D models is useful when the word vector semantic embedding are used during the training stage. Our proposed framework addressing the ZSL task is shown to produce good results on a number of benchmark datasets. In this paper, our main contributions are as follows:

- We present a new challenge to the ZSL community which aims to classify 3D point cloud objects without having previously observed a single instance of the classes they belong to.
- We adapt established 3D point cloud classification methods to perform ZSL which can serve as baseline performance for further research in this direction.
- We introduce a new evaluation protocol for ZSL methods on 3D point clouds which consists of a seen and unseen split of data from the datasets ModelNet40 [29], ModelNet10 [29], Mcgill [21] and SHREC2015 [12].

## 2 Related Work

**3D point cloud object recognition architecture:** The early methods utilizing deep learning for operating on 3D point clouds used volumetric [29] or multi-view [23] representations in order to work with 3D data. Recently, the trend in this area has shifted to instead using raw point clouds directly [18, 28, 10], without any preprocessing step. These methods do not suffer to the same degree from scalability issues as the volumetric representation does, and they do not make any *a priori* assumptions onto which 2D planes, and how many, that the point cloud should be projected on, like the view-based methods do. PointNet [17] was the first work that operated on raw point clouds directly at the input of the network. PointNet used a multi-layer perceptron (mlp) [20] to extract features from point sets, and max-pooling layers to remove the otherwise inherent issue of permutation from the point clouds. Later, many methods [18, 28, 10] were proposed to overcome the limitations of PointNet, which does not utilize local features or a more advanced pooling operation than max-pooling. At the time of writing, only the traditional recognition case where all the classes of interest have been seen at training time, have been considered in the case of 3D point cloud data. The current literature does not address the zero-shot version of the 3D recognition problem. In this paper, for the first time, we perform ZSL on 3D point cloud objects.

**Zero-shot learning on 2D images:** In the image recognition literature, zero-shot learning (ZSL) has made reasonable progress over the past few years [19, 32, 8]. The objective of such learning is to recognize objects from unseen classes not used during training. To do that, semantic information about the class labels in the form of attributes/word vectors are taken advantage of. Image features are usually transferred to the dimension of the semantic vector to obtain a matching score by comparing it with seen/unseen semantic vectors. Some of the notable research directions in this line of investigation include exploring class attribute association [3], domain adaptation [4], the effect of hubness [32], generalized ZSL [19], inductive vs. transductive ZSL [11], multi-label ZSL [9] etc. In this paper, we apply zero-shot learning on 3D point cloud objects instead of the traditional 2D image.

## 3 Our approach

**Problem formulation:** Suppose we define a set of seen $\mathbf{Y}^s = \{1, ..., S\}$ and a set of unseen $\mathbf{Y}^u = \{S+1, ..., S+U\}$ labels, where $\mathbf{Y}^s \cap \mathbf{Y}^u = \varnothing$, and $S$ and $U$ are the total number of seen and unseen labels respectively. There are associated semantic class embeddings (word vectors) for all samples in both seen and unseen sets, which is defined as $\mathbf{E}^s = \{\mathbf{e}_s : s \in \mathbf{Y}^s\}$ and $\mathbf{E}^u = \{\mathbf{e}_u : u \in \mathbf{Y}^u\}$ respectively, where $\mathbf{e}_s, \mathbf{e}_u \in \mathbb{R}^d$. The number of instances for sets of seen and unseen classes are $n_s$ and $n_u$ respectively. The matrices $\mathbf{X}^s = \left[\mathbf{x}_s^1, ..., \mathbf{x}_s^{n_s}\right]$ for $s \in \mathbf{Y}^s$, and $\mathbf{X}^u = \left[\mathbf{x}_u^1, ..., \mathbf{x}_u^{n_u}\right]$ for $u \in \mathbf{Y}^u$ are point cloud features for the seen and unseen classes respectively, where for both seen and unseen instances $\mathbf{x} = \{\mathbf{p}_1, ..., \mathbf{p}_n\}$ is an unordered point set, where $\mathbf{p}_i \in \mathbb{R}^F$. In the simplest setting of $F = 3$, each point contains 3D coordinates $x$, $y$, and $z$. To place the problem in a zero-shot setting, it is crucial to mention that, $\mathbf{X}^u$, $\mathbf{Y}^u$ and $\mathbf{E}^u$ are not observed during the training stage. Here, we define the ZSL task addressed in this work: Only point cloud features of seen classes $\mathbf{X}_s$ are used in the training phase. The aim is to assign an unseen class tag $u \in \mathbf{Y}^u$ to a given unseen 3D point cloud shape using the related feature vector $\mathbf{x}_u$.

**Training:** Given an unordered point set representing an object from a seen class $\mathbf{x}_s = \{\mathbf{p}_1, ..., \mathbf{p}_n\}$, a set function is defined such that any permutation of the point set becomes irrelevant,

$$f(\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n) \approx g(h(\mathbf{p}_1, \beta)), h(\mathbf{p}_2, \beta)), ..., h(\mathbf{p}_n, \beta))$$

where $f$ is the set function, $h$ is the feature extraction function, $g$ is the pooling function with the ability to remove the effects of permutation of points in a set, and $\beta$ represents a set of arguments associated with $\mathbf{p}_i$. The feature extraction function $h(\mathbf{p}_i, \beta)$ extracts a richer representation from the point cloud in a higher dimension. For feature extraction, two different algorithms for feature extraction, $h(\mathbf{p}_i, \beta)$, are applied. That is, global feature extraction as done in PointNet [17], and local feature extraction as done in EdgeConv [28]. In

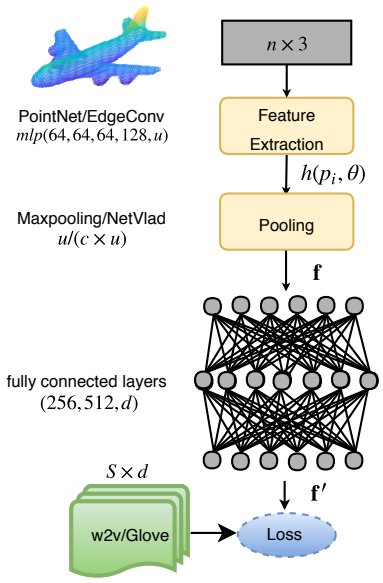


Figure 2. A general framework of the architecture using semantic word vectors. Similar to a traditional 3D point cloud recognition system, it produces a prediction score for every seen object. The inference of unseen classes is made based on those seen predictions.

PointNet, $h(\mathbf{p}_i, \beta) = h(\mathbf{p}_i) : \mathbb{R}^d \rightarrow \mathbb{R}^{d'}, \beta = \{\emptyset\}$, since each point is considered separately, the extracted feature vector contains global information. In EdgeConv [28], which extracts local features as well as global features, $h(\mathbf{p}_i, \beta) = h(\mathbf{p}_i, \mathbf{p}_j - \mathbf{p}_i) : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^{d'}, \beta = \{\mathbf{p}_j - \mathbf{p}_i\}$. In this case, point sets are represented by a dynamic graph and edge features based on $k$-nearest neighbors are calculated. Since point sets are inherently unordered, a function which is invariant to permutation is necessary to pool point features into a feature vector. Here, the Maxpooling operation above, $g$, is capable of removing the permutation from point clouds. Also, instead of Maxpooling, [24] used NetVlad [1] as a pooling mechanism to remove permutation from point cloud features.

Finally, via a collection of $h(\mathbf{p}_i, \beta)$, corresponding values of $f$ can be computed to form a vector $\mathbf{f} \in \mathbb{R}^m$ for Maxpooling pooling, and $\mathbf{f} \in \mathbb{R}^{m \times c}$ for NetVlad pooling, where $c$ is the number of centers in NetVlad, which represents a feature vector of an input point set. The obtained feature vector removes permutation from the point cloud. In the next step, a few fully-connected layers are applied to the feature vector $\mathbf{f}$ in order to transform the features into a more discriminative representation, $\mathbf{f}' = \phi(\mathbf{f})$, where $\phi : \mathbb{R}^m \rightarrow \mathbb{R}^d$ represents three fc layers with a nonlinear relu activation. This function maps the point cloud feature vector $\mathbf{f}$ to a semantic word vector embedding space by calculating $\mathbf{f}'$. Then, semantic embedding vectors $\mathbf{E}^s$, from all seen classes, are inserted into the network by multiplying with the feature vector $\mathbf{f}'$ from the fully connected layers, $\mathbf{f}'' = \mathbf{E}^s.\mathbf{f}'$, where $\mathbf{E}^s \in \mathbb{R}^{S \times d}$, and $\mathbf{f}'' \in \mathbb{R}^S$. Finally, the following objective function is minimized in order to train the proposed method shown in Figure 2:

$$L = -\sum_{i=1}^{S} Y_i^s log(y_i)) \quad \text{where}, y_i = \frac{e^{(E^s.f'_i)}}{\sum_{j=1}^{S} e^{(E^s.f'_j)}}$$

**Multiple semantic space fusion:** In Figure 2, only one semantic representation, w2v or glove, is considered during the training stage. However, it is also possible to fuse both semantic representations and consider them a unit semantic vector by concatenating the two different semantic embedding space representations. Therefore, the new semantic vector is: $E^s = concat(E^{s_{w2v}}, E^{s_{glove}})$, where $\mathbf{E}^{s_{w2v}} = \{\mathbf{e}_{s_{w2v}} : s \in \mathbf{Y}^s\}$, $\mathbf{E}^{s_{glove}} = \{\mathbf{e}_{s_{glove}} : s \in \mathbf{Y}^s\}$, and "*concat*" denotes the concatenation operator.

**Inference:** During the training stage, a classifier $p_s$ is trained on seen instances $\mathbf{X}^s$ such that it can estimate the probability of a 3D point cloud $\mathbf{x}$ belonging to a certain seen class label $s \in \mathbf{Y}^s$, denoted $p_s(s|\mathbf{x}_s)$, where $\sum_{s=1}^{S} p_s(s|\mathbf{x}_s) = 1$. Then, given $p_s$, we apply a method similar to [15], to transfer those probabilities obtained from training samples to the testing labels. Suppose $\hat{s}(\mathbf{x}, t)$ is the $t^{th}$ most likely label for a point cloud $\mathbf{x}$:

$$\hat{s}(\mathbf{x}, t) = \arg\max_{s \in Y^s} p_s(s|\mathbf{x}_s) \tag{1}$$

So, $p_s(\hat{s}(\mathbf{x}, t)|\mathbf{x})$ is the $t^{th}$ largest value in $\{p_s(s|\mathbf{x}); s \in Y^s\}$. Based on the top $T$ predictions of $p_s$, the proposed model calculates an embedding semantic feature vector $z(\mathbf{x})$ for an input $\mathbf{x}$,

$$z(\mathbf{x}) = \frac{1}{K} \sum_{t=1}^{T} p_s(\hat{s}(\mathbf{x}, t)|\mathbf{x}).e_{st} \tag{2}$$

where $T$ is a parameter in order to control the maximum number of embedding vectors that contributes in the inference stage, and $K = \sqrt{\sum_{t=1}^{T} (p_s(\hat{s}(\mathbf{x}, t)|\mathbf{x}))^2}$ is a normalization factor. The significance of using a subset of seen prediction values is to describe unseen classes via only closely related seen classes. Based on the predicted embedding of $\mathbf{x}$ in the semantic space, $z(\mathbf{x})$, zero-shot classification is applied to find the class which is nearest to the obtained embedding vectors. The top prediction for a point cloud $\mathbf{x}$ from the test set, denoted $\hat{y}(\mathbf{x}, 1)$, is defined by using the cosine similarity to rank the embedding vectors,

$$\hat{y}(\mathbf{x}, 1) = \arg\max_{u \in Y^u} cos(z(\mathbf{x}), E^u) \tag{3}$$

**An alternative approach to adapt ZSL:** In this paper, we modified the PointNet [17] and EdgeConv [28] architectures such that they can consider word vectors during training. This modification helps aligning the point cloud object features to their corresponding semantic space. The output of this architecture still provides a prediction value, $p_s(s|\mathbf{x}_s)$ for each seen class which we use during inference in Eq. 1. However, as the original PointNet [17] and EdgeConv [28] architectures can also provide such seen prediction values, one can consider the original framework instead of our modified architecture. Then, the inference follows the same process as mentioned above. We name this alternative process of prediction the *basic* approach. In

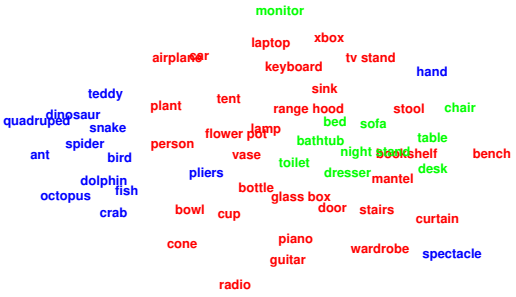


Figure 3. 2D tSNE [25] visualization of word2vec vectors [13]. Red, green and blue texts represent seen ModelNet40 [29], unseen ModelNet10 [29] and unseen McGill [21] classes respectively.

the experiments, we find that as this basic approach does not consider word vectors during training, it does not perform well predicting unseen classes (See Table 1). However, being free from the noise inside the word vectors, it performs very well recognizing objects of seen classes (See Table 2).

## 4 Experiment

### 4.1 Setup

**Dataset:** We perform our experiments on three well-known 3D datasets, ModelNet40/Modelnet10 [29], McGill [21], and SHREC2015 [12]. The ModelNet40 contains 12,311 CAD models from 40 different classes, and Modelnet10, which is a subset of ModelNet40, consists of 4899 CAD models from 10 different classes. The Modelnet40 contains 9,843 training samples and 2,468 testing samples, and ModelNet10 includes 3991 training samples and 908 testing samples. Moreover, the McGill dataset consists of 456 CAD models of 19 different objects. The SHREC2015 dataset consists of 1200 3D watertight triangle meshes from 50 different classes, where each class contains 24 objects with distinct postures. Each shape in the dataset has approximately 10,000 vertices. Note that, we randomly sample all models to 1,024 points from the mesh faces and normalize them to a unit sphere. We use the (x, y, z) coordinates of the sampled points for all the experiments in this work.

**Seen/Unseen split:** We are proposing a suitable and standard split of seen and unseen classes similar to the 2D version of the corresponding ZSL problem. In the 2D case, experiments using established methods of ZSL are performed on a single, well defined, split of seen/unseen classes from datasets like Animals with Attributes (AwA) [7], and Caltech-UCSD Birds (CUB) [26]. The idea of using a single split is better than using several random splits as it establishes a common testbed for evaluating other methods. In a real life setting, unseen 3D point cloud objects are likely obtained from an advanced 3D-depth camera or laser scanner. Here, however, we attempt to simulate

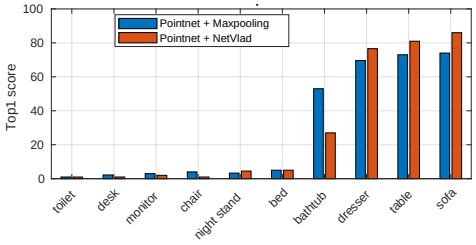


Figure 4. Per-class recognition rate of unseen ModelNet10 classes.

that scenario by using 3D point cloud objects from a different dataset not included during training. Therefore, we propose to use 30 classes from ModelNet40 as seen and other, disjoint classes, from ModelNet10, McGill and SHREC2015 as unseen. In ModelNet10, which has 10 classes, only testing samples based on the splitting protocol mentioned above, are considered as the unseen set. In McGill [21], 5 classes, which also appear in the seen set, are removed, and the remaining 14 categories are selected. Since we follow the protocol introduced by [27], we only consider their test set, the last third of the instances, as the unseen instances to enable a fair comparison. Finally, in the SHREC2015 [12] dataset, those classes that were similar to the seen classes, and those classes for which there were no semantic word vector available, were removed. The remaining 30 of the original 50 classes were chosen as the unseen instances. Moreover, in line with the protocol used in [31], 25% of instances were chosen randomly as unseen samples.

**Semantic features:** We work with unsupervised word vectors obtained from an unannotated text corpus as semantic information. We used $\ell_2$ normalized, 300 dimensional word2vec [14] and Glove [16] word vectors. The 2D tSNE visualization [25] of those vectors is illustrated in Figure 3.

**Evaluation:** The recognition performance is here measured by top-1 accuracy. It means the class with the highest predicted probability must match the ground truth class to be considered "correct". **Implementation details:** We use the Adam optimizer and a batch size of 16. We also use Relu and Batch normalization(BN) [5] for each layer. We implemented the architecture using TensorFlow and executed on an NVIDIA GTX1080Ti. For PointNet, we employed five shared mlp layers (64,64,64,128,1024). Also, for EdgeConv, which consists of two feature extraction blocks, the first block used three shared mlp layers (64,64,64) and the second block consisted of a shared mlp of size (128). Then, a shared mlp of size 1024 was used to concatenate all features together. Finally, for aggregation, we also used either max pooling or NetVlad. In the case of max pooling, we got a feature vector of size 1024. In the case of NetVlad, the number of cluster centers was set to 128 and the last mlp layer was set to size 128, from which we got a feature vector of size $128 \times 128$.

Table 1. Overall Top-1 accuracy on unseen classes. Random accuracy is calculated by $\frac{100}{\#\ of\ unseen\ classes}$

| Method | ModelNet10 | | | | McGill | | | | SHREC2015 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | basic | w2v | glove | conc | basic | w2v | glove | conc | basic | w2v | glove | conc |
| PointNet[17]+ Maxpooling | 23.1 | 27.0 | 14.8 | 19.4 | 14.1 | 9.8 | 7.2 | 11.6 | 3.1 | 4.1 | 3.6 | 4.2 |
| PointNet[17] + NetVlad[1] | 9.1 | **28.0** | 20.9 | 20.5 | 4.5 | 10.7 | 10.7 | **16.1** | 3.1 | 5.2 | 4.2 | **6.8** |
| EdgeConv[28] + Maxpooling | 21.2 | 24.4 | 14.1 | 16.9 | 12.3 | 8.9 | 9.8 | 8.0 | 3.3 | 6.2 | 4.7 | 5.2 |
| EdgeConv[28] + NetVlad[1] | 9.4 | 14.6 | 12.7 | 18.5 | 6.5 | 9.6 | 7.8 | 8.1 | 3.0 | 5.2 | 4.1 | 4.1 |
| Random | 10.0 | | | | 7.1 | | | | 3.3 | | | |

Table 2. Overall Top-1 accuracy on seen classes.

| Method | basic (40) | basic (30) | w2v (10) | glove (10) |
|---|---|---|---|---|
| PointNet[17]+Maxpooling | 89.2 | 89.5 | 87.4 | 87.6 |
| PointNet[17]+NetVlad[1] | 87.1 | 87.7 | 81.2 | 81.2 |
| EdgeConv[28]+Maxpooling | 92.2 | 92.3 | 90.7 | 89.5 |
| EdgeConv[28]+NetVlad[1] | 91.2 | 91.4 | 86.0 | 83.8 |

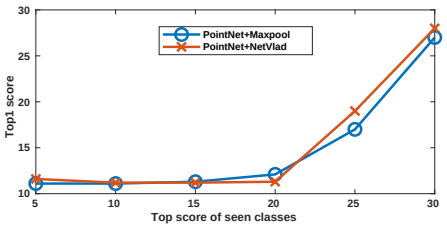


Figure 5. The effect of varying $T$, the number of embedding vectors in Eq. 2.

### 4.2 Recognition results

**Unseen recognition:** In this subsection, the top-1 accuracy performance of four different structures based on the combinations of feature extraction modules PointNet and EdgeConv, and the pooling layers Maxpooling and NetVlad, are evaluated. Moreover, for each structure, four different experiments, *basic, w2v, glove,* and *conc*atenation of w2v and glove, are conducted. It is important to mention that in the basic experiments, non semantic word vector embeddings are used during the training stage. As shown in Table 1, the winning architecture for point cloud zero-shot learning is the combination of PointNet and NetVlad, which achieves an accuracy of 28.9%, 16.1%, and 6.8% for the ModelNet10, McGill, and SHREC2015 datasets respectively. Based on the obtained results, we can make the following observations: 1) PointNet, in comparison to EdgeConv, has the advantage and performs better on the ZSL task, which can be due to the fact that the vanilla PointNet version has a simpler structure and fewer parameters to be learned. However, the EdgeConv module is better suited to the non-ZSL, point cloud recognition task than PointNet. 2) NetVlad works better than Maxpooling. Although Maxpooling is robust to permutation, it is a greedy operation which removes much useful information. In comparison to the Maxpooling layer, NetVlad keeps more features and, similarly, it ignores permutation in point cloud data. 3) Zero-shot 3D object recognition benefits from using a semantic word vector embedding in all investigated structures in comparison to the basic model. 4) Concatenation of w2v and glove helped to improve the performance of the proposed method on the McGill and SHREC2015 datasets.

**Per-class results:** In Figure 4, we report individual class performance of unseen ModelNet10 classes using PointNet + Maxpooling/NetVlad. Our architecture performs better on ModelNet10 classes (e.g., sofa, table, dresser, and bathtub) for which there are sufficiently similar classes in the seen set. However, one can find that the architecture cannot classify some unseen classes at all (for example toilet, crab, dinosaur etc). This is due to the hubness problem in high dimensional space Zhang *et al.* [32].

**Seen recognition:** To investigate the effect of using semantic feature vectors during the training stage, in the case of the traditional non-ZSL task, we also evaluated seen class performance (see Table 2). To this end, the test set of seen classes of the ModelNet40 splitting protocol are considered. Also, we tested basic models with 30 and 40 seen classes. However, the performance of the basic model is slightly better than the models which used semantic word vectors during training. Reasonably, being obtained in an unsupervised way, word vectors insert noise in the training which reduces the seen recognition rate but helps the unseen recognition rate.

**Parameter Dependency:** Eq. 2 allows us to vary the number of embedding vectors from the seen classes that contribute in the inference stage. In Figure 5, we see the result when varying this number from 5 to 30. Note that, in Fig. 3, the ModelNet10 vectors reside within the distribution of the ModelNet40 vectors . Hence, most seen ModelNet40 vectors contribute to describing unseen ModelNet10 objects. Therefore, ModelNet10 performs better with a large number of seen vectors.

## 5 Conclusion

Traditional recognition systems have achieved superior performance on 3D point cloud objects. However, due to the advancement of 3D depth camera technology, obtaining 3D point cloud representations of scenes has become much more accessible than before. Hence, we will more than likely encounter many unseen objects to which our traditional 3D point cloud recognition gets no training. Therefore, it is time for 3D point cloud recognition systems to adapt zero-shot settings,

aiming to recognize those unseen objects. To this end, this paper proposed a new challenge and a useful evaluation testbed/protocol for pushing forward with this new line of investigation. We also modified some established 3D point cloud recognition systems to work in the zero-shot setting in order to report a set of baseline performance results with respect to this problem. Overall, we believe that this research has the potential to motivate numerous further works to create a more robust 3D point cloud recognition system.

## References


[1] R. Arandjelovic, P. Gronát, A. Torii, T. Pajdla, and J. Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5297–5307, 2016.

[2] C. Chen, B. Yang, S. Song, M. Tian, J. Li, W. Dai, and L. Fang. Calibrate multiple consumer rgb-d cameras for low-cost and efficient 3d indoor mapping. *Remote Sensing*, 10(2), 2018.

[3] B. Demirel, R. Gokberk Cinbis, and N. Ikizler-Cinbis. Attributes2classname: A discriminative model for attribute-based unsupervised zero-shot learning. In *ICCV*, Oct 2017.

[4] S. Deutsch, S. Kolouri, K. Kim, Y. Owechko, and S. Soatto. Zero shot learning via multi-scale manifold regularization. In *CVPR*, July 2017.

[5] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[6] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 559–568, New York, NY, USA, 2011. ACM.

[7] C. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR Workshops*, pages 951–958, 2009.

[8] C. H. Lampert, H. Nickisch, and S. Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3):453–465, March 2014.

[9] C.-W. Lee, W. Fang, C.-K. Yeh, and Y.-C. Frank Wang. Multi-label zero-shot learning with structured knowledge graphs. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[10] J. Li, B. M. Chen, and G. H. Lee. So-net: Self-organizing network for point cloud analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9397–9406, 2018.

[11] Y. Li, D. Wang, H. Hu, Y. Lin, and Y. Zhuang. Zero-shot recognition using dual visual-semantic mapping paths. In *CVPR*, July 2017.

[12] Z. Lian, J. Zhang, S. Choi, H. ElNaghy, J. El-Sana, T. Furuya, A. Giachetti, R. A. Guler, L. Lai, C. Li, H. Li, F. A. Limberger, R. Martin, R. U. Nakanishi, A. P. Neto, L. G. Nonato, R. Ohbuchi, K. Pevzner, D. Pickup, P. Rosin, A. Sharf, L. Sun, X. Sun, S. Tari, G. Unal, and R. C. Wilson. Non-rigid 3D Shape Retrieval. In *Eurographics Workshop on 3D Object Retrieval*. The Eurographics Association, 2015.

[13] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, January 2013.

[14] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119. 2013.

[15] M. Norouzi, T. Mikolov, S. Bengio, Y. Singer, J. Shlens, A. Frome, G. S. Corrado, and J. Dean. Zero-shot learning by convex combination of semantic embeddings. In *ICLR*, 2014.

[16] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543, 2014.

[17] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017.

[18] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017.

[19] S. Rahman, S. Khan, and F. Porikli. A unified approach for conventional zero-shot, generalized zero-shot, and few-shot learning. *IEEE Transactions on Image Processing*, 27(11):5652–5667, Nov 2018.

[20] F. Rosenblatt. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books, Washington, 1962. it Early work on what would now be referred to as a "connectionist" model.

[21] K. Siddiqi, J. Zhang, D. Macrini, A. Shokoufandeh, S. Bouix, and S. Dickinson. Retrieving articulated 3-d models using medial surfaces. *Mach. Vision Appl.*, 19(4):261–275, May 2008.

[22] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[23] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015.

[24] M. A. Uy and G. H. Lee. Pointnetvlad: Deep point cloud based retrieval for large-scale place recognition. *arXiv preprint arXiv:1804.03492*, 2018.

[25] L. Van Der Maaten. Accelerating t-sne using tree-based algorithms. *Journal of machine learning research*, 15(1):3221–3245, 2014.

[26] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[27] C. Wang, B. Samari, and K. Siddiqi. Local spectral graph convolution for point set feature learning. *arXiv preprint arXiv:1803.05827*, 2018.

[28] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph cnn for learning on point clouds. *arXiv preprint arXiv:1801.07829*, 2018.

[29] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[30] Y. Xian, Z. Akata, G. Sharma, Q. Nguyen, M. Hein, and B. Schiele. Latent embeddings for zero-shot classification. In *CVPR*, June 2016.

[31] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. *arXiv preprint arXiv:1803.11527*, 2018.

[32] L. Zhang, T. Xiang, and S. Gong. Learning a deep embedding model for zero-shot learning. In *CVPR*, July 2017.