arXiv:2108.06230v3 [cs.CV] 19 Oct 2021

Generative Zero-Shot Learning for Semantic Segmentation of 3D Point Clouds

Björn Michele[1] Alexandre Boulch[1] Gilles Puy[1] Maxime Bucher[1] Renaud Marlet[1, 2]

[1]Valeo.ai, Paris, France [2]LIGM, Ecole des Ponts, Univ Gustave Eiffel, CNRS, Marne-la-Vallée, France

Abstract

While there has been a number of studies on Zero-Shot Learning (ZSL) for 2D images, its application to 3D data is still recent and scarce, with just a few methods limited to classification. We present the first generative approach for both ZSL and Generalized ZSL (GZSL) on 3D data, that can handle both classification and, for the first time, semantic segmentation. We show that it reaches or outperforms the state of the art on ModelNet40 classification for both inductive ZSL and inductive GZSL. For semantic segmentation, we created three benchmarks for evaluating this new ZSL task, using S3DIS, ScanNet and SemanticKITTI. Our experiments show that our method outperforms strong baselines, which we additionally propose for this task.

1. Introduction

3D perception is a critical part of many applications. We consider here two perception tasks on 3D point clouds: classification and, more importantly, semantic segmentation. The state of the art for these tasks is currently achieved by deep nets trained under full supervision. Yet, while 3D sensors have become more affordable, labeling 3D data has remained costly and time consuming. Semantic segmentation datasets for point clouds therefore contain a limited number of object and scene classes, with little intra-class variation, thus only covering partial real world situations. An option to address these limitations is to try to make predictions at inference time for objects unseen at training time, based on auxiliary information regarding non-annotated classes.

Zero-Shot Learning (ZSL) only predicts classes unseen at training time; Generalized ZSL (GZSL) predicts both seen and unseen classes. More precisely, while transductive (G)ZSL allows unlabeled objects of unknown classes to be part of training data, inductive (G)ZSL forbids it, making objects of unknown classes totally new to the model.

Much progress has been made on ZSL for image classification [60, 81, 75] and, recently, semantic segmentation [11, 80, 31, 42, 14]. But ZSL for point clouds has only been investigated for classification and by few studies [15, 18, 16]. We present here the first (to our knowledge) 3D GZSL approach for semantic segmentation.

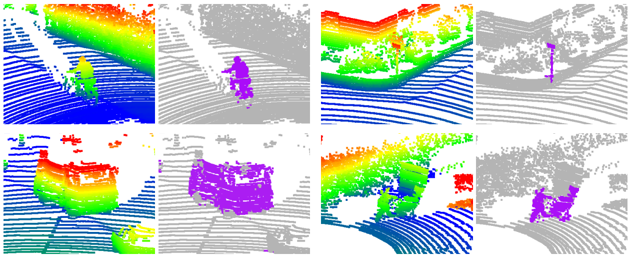


Figure 1. Zero-shot point cloud segmentation on SemanticKITTI. Point cloud with color gradient according to height (left image) and ZSL segment focusing on an unseen class (right) for classes bicyclist, traffic sign (top row), truck and motorbike (bottom row)

Classification makes sense for individual objects that are more or less isolated or centered. But except when making a digital 3D copy of an object (an easy labeling time), scans often observe a complex scene rather than a single object. 3D classification remains however relevant in the object detection task, when applied to a region proposal. We believe ZSL semantic segmentation is a more realistic scenario, applying to complex outdoor or indoor scenes as scanned by lidars or range cameras. It can be useful in particular as a 3D pre-annotation tool, e.g., for autonomous driving where country-specific objects (vehicles, roadsides, work barriers, possible road obstacles. . .) have to be widely collected and labeled. More precisely, the most relevant task is not ZSL but GZSL semantic segmentation, as it makes more sense not to forget about known classes when pre-annotating. Zero-shot segmentation can also be useful in the few-shot scenario as a way to mine large point cloud datasets to retrieve some examples to be manually annotated.

Our contributions are as follows. (1) We propose a generative framework handling both ZSL and GZSL for 3D point clouds, both for semantic segmentation and classification. (2) We make available 3 benchmarks for 3D GZSL semantic segmentation based, indoors, on S3DIS [2] and ScanNet [19], and outdoors, on SemanticKITTI [3] (cf. Fig. 1). (3) We define 2 baselines for 3D GZSL segmentation, which our method outperforms on the benchmarks.

2. Related work

2.1. Zero-shot learning for images

ZSL can be viewed as a special case of transfer learning, where knowledge from a source domain (seen classes) and source task (classification or segmentation) is transferred to a target domain (unseen classes) with a target task (different label space) [60, 81, 75]. We review some existing methods.

Attribute classifier. A first class of methods aims to recognize new objects based on attributes [38, 21, 47, 33, 1]. Here, no attribute list is available. But methods like ConSE [53] can also use word embeddings as attributes and can thus be adapted for ZSL on 3D point clouds [18].

Projection methods. This class of methods creates a mapping between an object representation and auxiliary data (class prototypes) such as word embeddings, e.g., W2V [52] or GloVe [55]. Class are then assigned in the prototype space [22, 78, 68]. However, these methods often face the hubness problem [59], where a (hub) class prototype is the nearest neighbor of a large number of other prototypes. To tackle this problem, an alternative is to do the comparison in the object representation space [66, 85].

Generative models. ZSL can be seen as a missing data problem: no examples of unseen classes are available at training time. *Generative methods* create artificially this missing data to train a classifier under supervision [10, 82]. As in [22], a CNN extracts visual features of seen classes, which are used to train the generative module, conditioned on the corresponding class prototype. Generative models are known to reduce the bias towards seen classes in GZSL and often to be superior to projection methods [65]. A great variety of generative modules may be used to create artificial features. Adversarial auto-encoders [50], conditional generative adversarial models [54], denoising auto-encoders [4] or GMMNs [43] are used in [10]. Wasserstein GANs [27] have been used in [82], and VAEs in [37, 83].

While f-CLSWGAN [82] focuses on the GAN aspect to make generated features somehow look more realistic and only uses the classification loss as a regularizer, our feature generation is only driven by classification (as [11]), which is the task target anyway. We thus save the tuning of 2 extra hyperparameters and we do not face the difficulty to train a GAN, as recognized in [82]. Besides, conditioning a discriminator on word embeddings is probably harder than on the simple attributes used in [82] itself. Moreover, training a discriminator to make generated features look like real features may be harmful with 3D datasets, that offer less training data than image datasets, thus less samples to learn real-looking features. The fact is we significantly outperform the adpatation of f-CLSWGAN 3D (cf. Table 1).

Semantic segmentation. While ZSL for image classification has been studied extensively (see above), semantic segmentation has only recently been tackled. [86, 34] focus on the discovery of objects of interest in a scene, either using a hierarchical open vocabulary approach [86] or splitting semantic segmentation into a foreground/background segmentation step and a classification step [34].

Other methods address the problem in a GZSL setting. [80, 31] project the object representation onto the class prototype space using semantic projection layers. On the contrary, [11, 26, 42, 14] project class prototype representations onto the object representation space and generate pixel-wise features of seen and unseen classes that are used to train a classifier. Our generative method belongs to this second group of approaches, adapting them to the special case of 3D point clouds.

2.2. Point cloud analysis with deep learning

Simple ways to adapt 2D methods to 3D data include conversion to range images [28, 49], image generation from virtual viewpoints [69, 6, 40], projection on 2D planes [70] and using voxel grids [51, 62, 57, 77, 61, 25]. Graph Neural Networks (GNNs) have been used to limit the loss of information due to data projection. They operate on graphs based on 3D neighborhoods [64, 8], possibly pre-segmented [39], using message passing [24, 44] or defining convolution in the spectral domain [9, 20, 36]. Deep learning on raw point clouds has now become commonplace. The points can be processed all together, like in PointNet [56], or using a hierarchical structure [58, 41, 32, 71, 5, 84, 74, 48, 76].

2.3. Zero-shot learning for 3D point clouds

To our knowledge, only 4 publications study ZSL for point clouds [18, 15, 16, 17], and they only address the classification task. The pioneering work [18] adapts ConSE [53] to 3D, using PointNet [56] to create an object representation, and GloVe [55] or W2V [52] as auxiliary information. [15] reduces the hubness problem of [18] using a loss function composed of a regression term [85] and a skewness term [59, 66], and extends to GZSL. The transductive case is discussed in [16] which extends [15] using a triplet loss. Finally, the hubness problem is addressed in [17] along with the proposition of a unified approach for [18, 15, 16].

None of these approaches is generative. Yet, [16] transposes generative 2D methods [82, 65] to 3D for comparison. Poor results lead [16] to hypothesize they do not generalize well to 3D because their performance in 2D is mostly due to the high quality of pre-trained models (on millions of labeled images featuring thousands of classes), which do not exist for 3D data. In this work, we show that even with small datasets and a few classes, generative methods outperform the state of the art for 3D point clouds ZSL and GZSL, and also generalize to 3D semantic segmentation. Recently, semantic segmentation is discussed in the technical report [46], concurrent to our work. The setting used in [46] can be referred as zero-label learning [79], i.e., the unseen classes are present at training time. On the contrary, we follow a ZSL procedure, removing from the training set all point clouds containing unseen classes. Furthermore, we are also addressing in our method the bias problem, which semantic segmentation faces due to the appearance of seen and unseen classes in the same scene.

3. Method

Point cloud semantic segmentation can largely be seen as the classification of individual 3D points, although specific developments are required (see below). In this section, we introduce a general generative ZSL framework that applies both to classification and semantic segmentation.

3.1. Problem formulation

Let $\mathcal{C}$ be a set of object classes, $\mathcal{P} = (P_i)_{i \in I}$ be a set of objects, and $\mathcal{Y} = (y_i)_{i \in I}$ be the set of corresponding class labels. For classification, an *object* P_i to label is a point cloud; for semantic segmentation, it is a 3D point.

The object classes $\mathcal{C}$ are partitioned into seen classes $\mathcal{S}$ and unseen classes $\mathcal{U}$. The objects of seen classes are $\mathcal{P}^{\mathcal{S}} = (P_i)_{i \in I^{\mathcal{S}}}$ with $I^{\mathcal{S}} = \{i \in I \mid y_i \in \mathcal{S}\}$, and likewise for unseen classes $\mathcal{U}$. At training time, only objects $(P_i)_{i \in I^{\mathcal{S}}}$ and corresponding class labels $(y_i)_{i \in I^{\mathcal{S}}}$ are available.

Class prototypes. Learning from $\mathcal{P}^{\mathcal{S}}$ and generalizing to $\mathcal{P}^{\mathcal{U}}$ without seeing any example from a class in $\mathcal{U}$ is impossible without extra knowledge. We have to rely on *auxiliary information*: the so-called *class prototypes*, which are not exemplars (as no “shot” is allowed) but D-dimensional embedding vectors. It is denoted by $\mathcal{T} = \{t_c \in \mathbb{R}^D \mid c \in \mathcal{C}\}$, where each class has a single prototype. We distinguish $\mathcal{T}^{\mathcal{S}}$ and $\mathcal{T}^{\mathcal{U}}$, the subsets of $\mathcal{T}$ for seen and unseen classes.

Object representations. Our objective is to embed the objects in $\mathcal{P}$ and the class prototypes in $\mathcal{T}$ into a common object representation space $\mathcal{X}$ where objects and prototypes of the same class have similar embeddings. The embedding function for objects is denoted by $\phi(\cdot)$ and is typically implemented using a deep neural network. Our embedding function for class prototypes is a generator denoted by $G(\cdot)$.

Training set. We consider the difficult case of *inductive* ZSL: no data on unseen classes is available at training time; only their class prototypes is available, at test time. The training set thus consists of the triplets $(P_i, y_i, t_{y_i})_{i \in I^{\mathcal{S}}}$ where $(t_{y_i})_{i \in I^{\mathcal{U}}}$ are the class prototypes of unseen classes.

Test set. We test on objects $\mathcal{P}_{\text{test}} = (P_i)_{i \in I_{\text{test}}}$ labeled in $\mathcal{Y}_{\text{test}} = (y_i)_{i \in I_{\text{test}}}$, where I_{test} indexes test samples. When $\mathcal{Y}_{\text{test}}$ contains only classes in $\mathcal{U}$, it is the *vanilla ZSL* test setting; when $\mathcal{Y}_{\text{test}}$ contains classes both in $\mathcal{S}$ and $\mathcal{U}$, it is the *generalized* ZSL test setting. Semantic segmentation mainly makes sense in complex scenes with several co-located objects of different classes. As in practice, seen and unseen classes will often simultaneously appear in a scene, we consider semantic segmentation only in the GZSL setting.

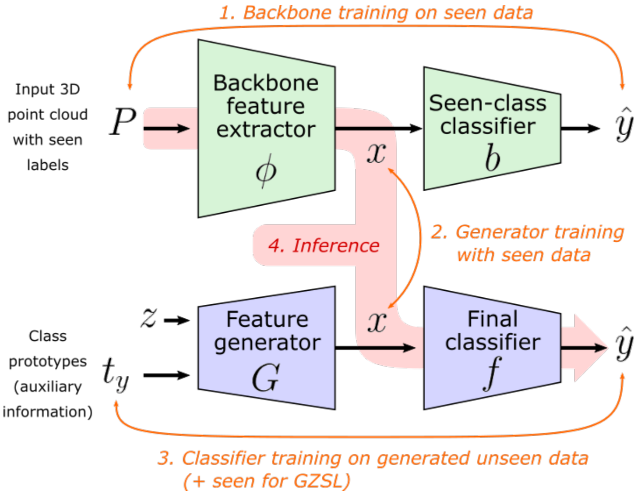

1. Backbone training on seen data
Input 3D point cloud with seen labels
P
Backbone feature extractor
ϕ
x
Seen-class classifier
b
$\hat{y}$
4. Inference
2. Generator training with seen data
z
Class prototypes (auxiliary information)
t_y
Feature generator
G
x
Final classifier
f
$\hat{y}$
3. Classifier training on generated unseen data
(+ seen for GZSL)

Figure 2. Four-step training and inference procedure: (1) backbone training on the seen classes, (2) generator training, (3) classifier training with artificial unseen features for ZSL (unseen and seen for GZSL), (4) inference through backbone and final classifier.

3.2. Our approach

Our approach relies on 3 main modules, trained sequentially (see Fig. 2): a backbone feature extractor $\phi(\cdot)$ processing 3D point clouds, a generative module $G(\cdot)$ taught to generate features x conditioned on a class prototype t, and a classifier $f(\cdot)$ predicting a class $\hat{y}$ given a feature x.

Backbone feature extractor. Each object P_i in $\mathcal{P}$ is represented by a set of 3D points $\bar{P}_i$. For classification, $\bar{P}_i$ is the point cloud P_i itself; for semantic segmentation, $\bar{P}_i$ is the point cloud containing the point P_i to label. The object embedding function $\phi(\cdot)$ is a backbone, such as PointNet [56]. It extracts one representation $x = \phi(P_i)$ for classification, or several (one for each point of $\bar{P}_i$) for segmentation.

We first train the feature backbone $\phi(\cdot)$ under full supervision of the seen classes, combining it with a linear classifier $b(\cdot)$. For each example $(P_i, y_i)_{i \in I^{\mathcal{S}}}$ of a seen class, we compare $b(\phi(P_i))$ to y_i via a cross-entropy loss to train both $b(\cdot)$ and $\phi(\cdot)$. Only $\phi(\cdot)$ is used afterwards; $b(\cdot)$ is ignored.

Feature generator. The role of generator $G(\cdot)$ is to create a training set of fake but realistic unseen object representations, to train a final classifier for unseen classes. These generated features should be similar to $\phi(P)$ for $P \in \mathcal{P}^{\mathcal{U}}$, i.e., to features obtained if we had had access to unseen data. For this, we use a generative approach like in [10, 11]. Unlike ϕ, that is deterministic, features of class prototypes are generated as $x = G(\mathbf{z}_j, t_s)$, where $\mathbf{z}_j$ is a random vector and $t_s \in \mathcal{T}^{\mathcal{S}}$is the prototype of a seen class $s \in \mathcal{S}$. $G(\cdot)$ is trained to generate features $x_i = G(\mathbf{z}_j, t_{y_i})$ similar to backbone features $\phi(P_i)$, from examples $(P_i, y_i)_{i \in I^{\mathcal{S}}}$ of seen classes.

Final classifier. The final feature classifier $f(\cdot)$ is trained with a cross-entropy loss. For ZSL, it supervised by generated features of unseen classes only, i.e., using a training set

$D^{\mathcal{U}} = \{(G(\mathbf{z}_j, t_y), y) \mid y \in \mathcal{U}, 1 \leq j \leq |D^{\mathcal{U}}|\}$. For GZSL, $f(\cdot)$ is trained on $D^{\mathcal{U}\cup\mathcal{S}}$, containing features generated for unseen classes and features of the seen training data (backbone outputs). Classification is successful if the distribution of generated features in $D^{\mathcal{U}}$ is similar to that of $\phi(P)$ for P in $\mathcal{P}^{\mathcal{U}}$ (or P in $\mathcal{P}^{\mathcal{C}}$ for GZSL) and if the object representation space can be linearly separated for each class.

Inference. For inference, given a point cloud P, features are first extracted using the trained backbone ϕ, then classified using the final classifier f (see Fig. 2). In other words, inference consists in computing $\hat{y} = f(\phi(P))$. Therefore, the inference time and space complexity basically is that of the backbone. The generator is only used at training time.

Reducing bias toward seen classes. In GZSL, both unseen and seen classes appear in the test set. As described in [13], a bias toward seen classes can then be observed. The reason is twofold. First, the feature extractor may focus only on features useful to discriminate seen classes, inducing a loss of information required to deal with unseen classes. Second, as the generator is trained only on seen classes, it generates feature of better quality for seen classes than unseen ones. This bias is addressed in two ways:

Class-dependent weighting. When we train the classifier, the loss for unseen classes is weighted with a factor $\beta > 1$. The assumption is that, as the generator is only trained on seen classes, it generates lower quality features for unseen classes, which are thus more difficult to classify. Therefore, we give more importance to the unseen classes at training time, forcing the classifier to focus on them.

Calibrated stacking [13]. The bias for seen classes can be reduced at test time as a post-processing by subtracting a small value ϵ from the seen-class score (after softmax).

The weight factor β and the offset ϵ are hyperparameters chosen using a validation set created from the train set.

4. Experiments

4.1. Datasets and Metrics

Classification. ModelNet40 [77] is used in [18, 15, 16, 17] as a classification benchmark. It consists of 40 object classes of CAD objects. For the ZSL setting, it is split into 30 seen and 10 unseen classes. The 10 unseen classes are the ones of ModelNet10, which is a subset of ModelNet40.

Semantic segmentation. As it is the first time ZSL semantic segmentation is tackled for 3D data, there is no reference benchmark. We created 3, based on 3 common 3D semantic segmentation datasets. S3DIS [2] includes point clouds of 271 scanned rooms, with points labeled among 13 classes. ScanNet [19] contains 1513 indoor scans with annotations for 20 classes. SemanticKITTI [3, 23] provides point clouds acquired by a lidar on a car driving in the streets. Grouping moving and non-moving objects with the same semantics results in 19 different classes. We keep the same 10 sequences for training but, as test sequences are unavailable, we use the validation sequence for ZSL testing.

ZSL splits. To assess ZSL, we need seen and unseen classes. To ease the adoption of our benchmark, we create a single but rich ZSL split per dataset, with a variety of difficulties, while allowing to leverage on textual semantic proximity. To make the datasets appropriate for the inductive setting, we discard, in the seen-class training set, any point cloud containing an instance of an unseen class. It is a hard but necessary constraint, although it reduces the size of the original datasets. The fact is keeping all point clouds in their entirety and learning only from labeled seen-class points cannot qualify as an inductive setting because the backbone can then leverage on the contextual presence of unlabeled unseen-class points, even if these unseen-class points do not back-propagate class information.

The small number of classes and their distribution reduce options as there must be enough samples of seen classes to train on. Yet to make segmentation challenging we consider 4 unseen classes in each dataset: *beam*, *column*, *window*, *sofa* for S3DIS; *desk*, *bookshelf*, *sofa*, *toilet* for ScanNet; *motorcycle*, *truck*, *bicyclist*, *traffic-sign* for SemanticKITI. For indoor scenes (S3DIS, ScanNet), expected semantically close categories are *sofa* (unseen) and *chair* (seen), and *desk* (unseen) and *table* (seen). For outdoors (SemanticKITTI), *poles* are chosen as seen while *traffic signs*, most of which are attached to a pole, are not. To allow inductive ZSL, traffic-sign poles are thus not seen at training time. Yet the split allows to evaluate the correlation. Moreover, *bicycles* (usually parked) are seen while *bicyclists* (including the bikes) are not; it allows to evaluate the ability to add an unseen rider onto a seen class, and conversely for *motorcycle* and *motorcyclist*. Last, *trucks* are unseen, with no direct correlation other than being vehicles like *cars*.

Metrics. We evaluate the methods with commonly used metrics: global accuracy (Acc) and accuracy per class for classification; average intersection-over-union (mIoU) for semantic segmentation. In the particular case of GZSL, as the results may be biased toward seen classes, a common metric is to report the Harmonic mean (HM) of the the measures for seen and unseen classes (whether Acc or mIoU).

4.2. Backbone feature extractors and generators

Feature extractors. Many backbones allow point cloud classification and segmentation. We experimented with 4, illustrating our method is not backbone-dependent. For classification, we chose PointNet as in [15, 18] to enable comparisons and to show our results are not just due to a better backbone. Besides, PointNet is only 4% less accurate than the state of the art on ModelNet40. For segmentation, in the absence of prior work, we chose three backbones at the state of the art when trained under full supervision: ConvPoint [5] for S3DIS, FKAConv [7] for ScanNet, and KP-

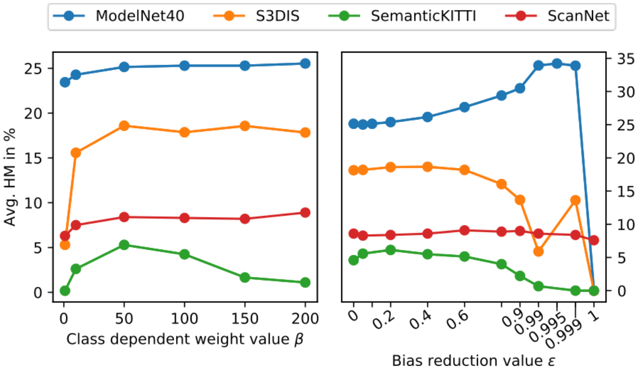


Figure 3. Effect of bias reduction parameters on HmAcc (ModelNet40 classification) and HmIoU (semantic segmentation on S3DIS, ScanNet, SemanticKITTI): (a) class-dependent weighting factor β, (b) calibrated stacking value ϵ (average over 20 runs).

Conv [71] for SemanticKITTI. Each backbone is trained on the seen classes using the recommended setting described in its respective paper.

Generators. We tested 4 generators, as in [10, 11]: a denoising auto-encoder (DAE) [4], a generative moment-matching network (GMMN) [43], a conditional GAN [54] and an adversarial auto-encoder [50]. We chose the best performing generator on the validation sets: DAE for classification and GMMN for segmentation (see supp. mat.).

4.3. Parameter setup

Parameters governing the training are selected by cross-validation. We create validation sets out of the training sets.

Validation sets. To create cross-validation splits, we follow the ZSL protocol of [10]: we randomly select 20% or at least 2 of the seen classes of the training data as validation classes. The feature backbone and the generator are trained from scratch on each split, using only the seen classes not selected for validation. Splits are evaluated only on validation classes. For semantic segmentation, some classes are present in almost every pillar (chunk to process for point clouds), e.g., floor or ceiling for indoor scenes; they cannot be selected as unseen validation class.

Number of generations. In ZSL, the final classifier $f(\cdot)$ is trained on artificial examples. For classification, a study of the impact of the number of generated examples at training allows to observe that: first, a very small dataset does not perform well and second, a plateau is reached at about 100 samples per class, with a maximum around 500, which is the value we select. For semantic segmentation, we follow the procedure from [11], where object representations are generated according to their frequency in the dataset.

Bias reduction in GZSL. As β is a training parameter and ϵ is used for post-processing, we tune them sequentially with cross-validation. A range study is shown on Fig. 3.

Class-dependent weighting. We observe on Fig. 3(a) the relative stability of the evaluation metric on all datasets/ tasks when $10 < \beta < 100$. As a close-to-maximum value is reached for $\beta = 50$, we use this value in all experiments.

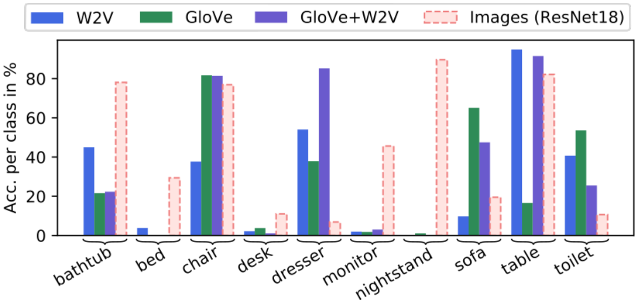


Figure 4. Per-class accuracy for each unseen class of ModelNet40, for each kind of embedding (word and image, see Sect. 4.5).

Calibrated stacking. We then select a value of ϵ. As the best ϵ varies substantially for each dataset (see Fig. 3(b)), we choose it dataset-wise: 0.995 for ModelNet40, 0.4 for S3DIS, 0.6 for ScanNet and 0.2 for SemanticKITTI.

Note that class-dependent weighting ($\beta > 1$) yields a gain up to 2 pts on classification (see Fig. 3(a), compared to $\beta = 1$) while calibrated stacking gains up to 8 pts. However, it brings a gain up to 13 pts on the segmentation task, which fully justifies the use of both bias reduction techniques.

4.4. Benchmark results

We now present the results of our method on the benchmarks. As opposed to the previous section, where parameter studies were made on small validation sets created out of the training sets, we train here on the whole training sets of seen classes, which explains the better results we obtain.

4.4.1 Classification

We evaluate classification on ModelNet40 [77] using classes of ModelNet10 as unseen, like [18], to allow comparison. Classification results for both ZSL and GZSL are presented in Table 1 (averaged over 20 runs for our method).

Influence of auxiliary information. Results are presented for three class prototypes: Word2Vec (W2V) [52], GloVe [55], and their concatenation (GloVe+W2V). For both ZSL and GZSL, we observe that, while using W2V or GloVe alone leads to similar performances, their concatenation performs much better: +7.5% (ZSL), +5.4% (GZSL).

Fig. 4 provides detailed ZSL accuracy for each unseen class and each kind of class prototype (see supp. mat. for GZSL). We see that some classes (*bed*, *desk*, *monitor*, *nightstand*) are incorrectly predicted (if predicted at all) regardless of the class prototype. Except for *bathtub* and *toilet*, W2V+GloVe outperforms the worst of the two embeddings by a large margin. No performance gain is however guaranteed by using W2V+GloVe over W2V or GloVe alone.

Method		Full	ZSL				GZSL								
		super-			Glove		W2V			GloVe			GloVe + W2V		
	Gener-	vision	W2V	GloVe	+ W2V	Bias	Acc.	Acc.	HM	Acc.	Acc.	HM	Acc.	Acc.	HM
	ative	Acc.	Acc.	Acc.	Acc.	reduct.	$\mathcal{S}$	$\mathcal{U}$		$\mathcal{S}$	$\mathcal{U}$		$\mathcal{S}$	$\mathcal{U}$	
PointNet [56]		89.2													
f-CLSWGAN* [82]	✓		20.7	-	-		76.3	3.7	7.0	-	-	-	-	-	-
CADA-VAE* [65]	✓		23.0	-	-		84.7	1.3	2.6	-	-	-	-	-	-
ZSLPC [18]			28.0	20.9	20.5		40.1	22.5	28.8	49.2	18.2	26.6	-	-	-
MHPC [15]			**33.9**	28.7	-	✓	**53.8**	26.2	35.2	**53.8**	25.7	**34.8**	-	-	-
3DGenZ (ours)	✓		28.6	**29.3**	**36.8**	✓	48.8	**29.3**	**36.6**	44.7	**28.4**	34.7	**47.8**	**36.5**	**41.3**

Table 1. ZSL and GZSL classification results (in %) on ModelNet40. *: adaptation of 2D methods to 3D point clouds, implemented in [16]. Results for ZSLPC are with the best reported variant, i.e., PointNet + NetVlad [18]. For fair comparison we report results with the same PointNet backbone. Results are averaged over 20 runs for our method.

	Training set		S3DIS				ScanNet				SemanticKITTI			
	Back-	Classi-	mIoU			HmIoU	mIoU			HmIoU	mIoU			HmIoU
	bone	fier	$\mathcal{S}$	$\mathcal{U}$	All		$\mathcal{S}$	$\mathcal{U}$	All		$\mathcal{S}$	$\mathcal{U}$	All	
Supervised methods with different levels of supervision														
Full supervision	$\mathcal{S}\cup\mathcal{U}$	$\mathcal{S}\cup\mathcal{U}$	74.0	50.0	66.6	59.6	43.3	51.9	45.1	47.2	59.4	50.3	57.5	54.5
ZSL backbone	$\mathcal{S}$	$\mathcal{S}\cup\mathcal{U}$	60.9	21.5	48.7	31.8	41.5	39.2	40.3	40.3	52.9	13.2	42.3	21.2
ZSL-trivial	$\mathcal{S}$	$\mathcal{S}$	70.2	0.0	48.6	0.0	39.2	0.0	31.3	0.0	55.8	0.0	44.0	0.0
Generalized zero-shot-learning methods														
ZSLPC-Seg* [18]†	$\mathcal{S}$	$\mathcal{U}$	65.5	0.0	45.3	0.0	28.2	0.0	22.6	0.0	49.1	0.0	34.8	0.0
DeViSe-3DSeg* [22]†	$\mathcal{S}$	$\mathcal{U}$	70.2	0.0	48.6	0.0	20.0	0.0	16.0	0.0	49.7	0.0	36.6	0.0
ZSLPC-Seg [18]†	$\mathcal{S}$	$\mathcal{U}$	5.2	1.3	4.0	2.1	16.4	4.2	13.9	6.7	26.4	10.2	21.8	14.7
DeViSe-3DSeg [22]†	$\mathcal{S}$	$\mathcal{U}$	3.6	1.4	3.0	2.0	12.8	3.0	10.9	4.8	42.9	4.2	27.6	7.5
3DGenZ (ours)	$\mathcal{S}$	$\mathcal{S}\cup\hat{\mathcal{U}}$	**53.1**	**7.3**	**39.0**	**12.9**	**32.8**	**7.7**	**27.8**	**12.5**	**41.4**	**10.8**	**35.0**	**17.1**

Table 2. GZSL semantic segmentation results (in %). †Our adaption of the method. *Direct, unrepaired (failing) adaptation.

Comparison with state-of-the-art methods. Tab. 1 reports scores for ZSLPC [18] and MHPC [15] as well as the adaptation to 3D point clouds of the 2D methods f-CLSWGAN* [82] and CADA-VAE* [65] proposed in [16]. All methods, including ours, use the same backbone.

For ZSL, our method performs the best with GloVe and places second with W2V. Interestingly, it establishes the new state of the art with W2V+GloVe, while the previous state-of-the-art method [18] shows a significantly lower accuracy when combining both embeddings. It could be due to the nearest-neighbor search in a higher dimension space.

For GZSL, our approach is less accurate than baselines on seen classes. It is due to the parameter setting policy of our bias reduction techniques, that we set to favor similar scores for seen and unseen classes, trading seen-class accuracy for more accurate unseen classes. The fact is we outperform on HmAcc the other methods for W2V, and reach the state of the art for GloVe. With W2V+GloVe, we outperform the state of the art by a significant margin.

This shows that, contrary to what was believed [16], 2D generative ZSL methods can successfully be transferred to 3D, and even outperform non-generative ZSL methods on point clouds. In particular, we invalidate the hypothesis that the success of 2D generative models relies on pre-trained models [16]: all our networks are trained from scratch, only on seen classes of moderately-sized 3D datasets. Transfer to 3D however is not straightforward. In particular, contrary to a number of other 2D ZSL methods, 2D generative approaches are known not to require reducing a bias towards seen classes, as they work at feature level and can generate as many unseen examples as needed. Yet, because 3D backbones trained from scratch are somehow too specialized on seen classes, compared to 2D backbones pre-trained on huge datasets, we had to resort to two bias reduction techniques to outperform the state of the art on 3D ZSL.

4.4.2 Semantic Segmentation

We now present results for semantic segmentation on S3DIS, ScanNet and SemanticKITTI. As we are the first to address this task, there is no method to compare with. We however define two baselines, before comparing them to our method. The scores are reported in Table 2.

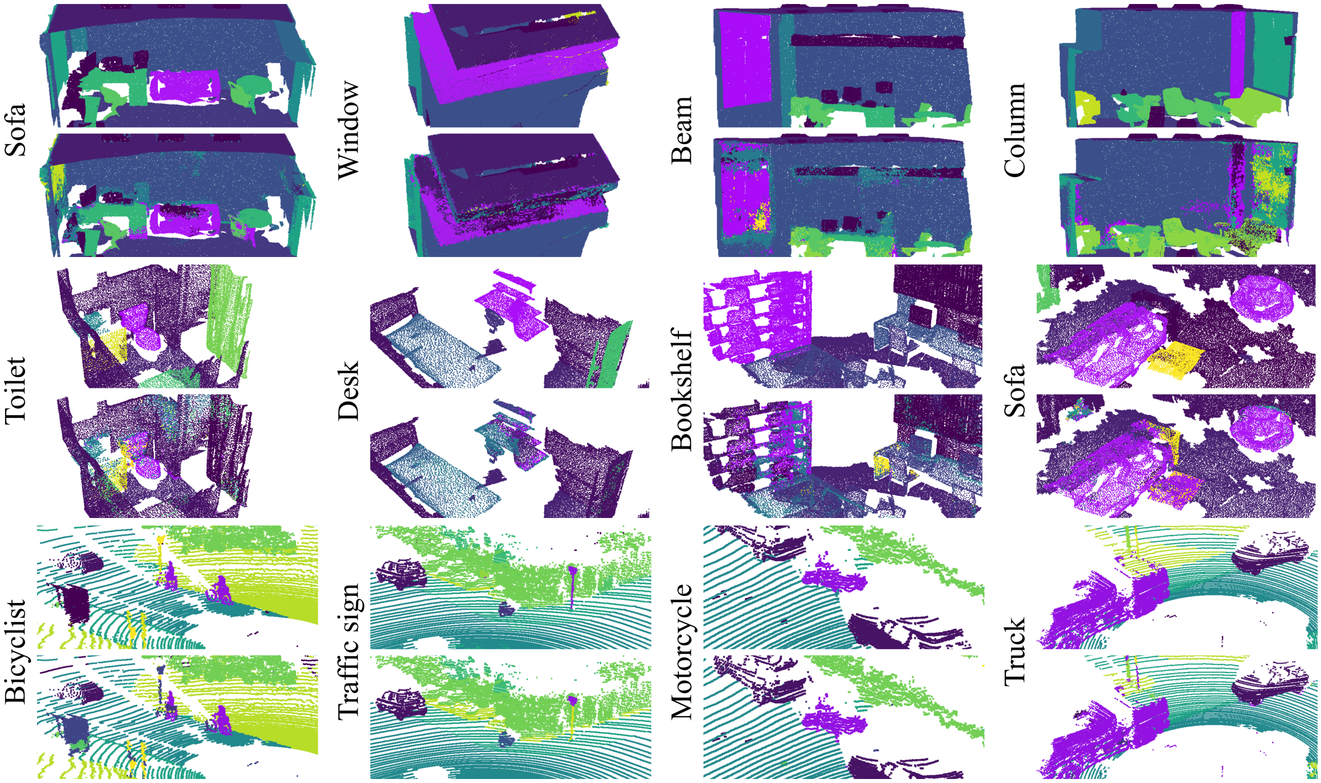


Figure 5. Zero-shot segmentation on scenes from S3DIS (row 1-2), ScanNet (row 3-4) and SemanticKITTI (row 5-6). For each two-row block, first row is the ground truth, second row is our ZSL prediction. Unseen classes have pink colors.

Supervision as upper bound. To scale our results, we train three supervised models (see grey rows in Tab. 2): with the backbone feature extractor and the classifier trained on all classes, seen and unseen (full supervision); with the backbone trained on seen classes only, and the classifier on all classes (ZSL backbone); with the backbone and classifier trained on seen classes only (ZSL-trivial). The first model is an upper bound when all classes are seen; the second provides an upper bound for zero-shot methods. It performs noticeably worse than the fully-supervised model, which hints that the backbone feature extractor trained only on seen classes generates object representations for unseen classes that are not easily distinguishable from seen classes.

For seen classes, our method behaves similarly with all datasets: it reaches a mIoU around 10 points below the maximum score (reached by ZSL backbone and a fully-supervised classifier). However, we notice different a behaviour on the unseen classes. While on SemanticKITTI, our performance is close to the performance of the ZSL backbone, we observe a much larger gap on S3DIS and ScanNet. This suggests that the generation of class prototypes for the unseen classes is of lower quality on S3DIS and ScanNet than on SemanticKITTI.

Baselines. We create two baselines for zero-shot 3D segmentation: (1) ZSLPC-Seg is an adaptation of the ZSL classification method ZSLPC [18] to segmentation. ZSLPC is itself an adaptation of ConSe [53] from 2D images to 3D point clouds. (2) DeViSe-3DSeg is an adaptation of Devise-Seg to 3D point clouds, Devise-Seg being itself an adaptation of DeViSe [22] from classification to segmentation, as proposed in [11]. The two baselines rely on a nearest-neighbor search, either in the class prototype space or in the object representation space. However, experimentally, searching the $K=1$-nearest neighbors leads to no prediction at all for unseen classes (light-red rows in Table 2). It illustrates that adapting a ZSL method to 3D or to segmentation is not straightforward. To construct meaningful baselines, we modify these methods by looking for the nearest unseen class among the K-nearest neighbors. If no such unseen class is found, then the class of the closest neighbor is selected. The values of K that maximize the HmIoU are: for DeViSe-3DSeg, $K=7,2,5$ on S3DIS, ScanNet, SemanticKITTI, respectively; for ZSLPC-Seg, $K=5,2,5$.

Results. Our approach outperforms the baselines by large margins and establishes the state of the art on the three defined benchmark. According to the HmIoU metric, the gap is much larger on S3DIS than on SemanticKITTI and ScanNet. It is possible that our method deals relatively better with a smaller number of classes, while the baseline methods benefit more from a larger number of classes.

Figure 5 provides qualitative results of our segmentation method on the three datasets. On S3DIS, the object are correctly located, but the classifier hesitates between classes, e.g., between window and board. As a result, predictions are mixed, which produces a salt-and-pepper effect. On ScanNet, the unseen classes *sofa* and *toilet* seem well segmented, with a bit of oversegmentation on *sofa*. On the contrary, the network has more difficulties to segment *desks* and *bookshelves*. On SemanticKITTI, objects appear much better segmented even though the mIoU for unseen classes in Table 2 is close to the one on S3DIS. The network is more confident, resulting in consistent segmentations, but it oversegments some classes. The semantic similarity between a pole and a traffic-sign, and between a two-wheeler with or without a rider, is a useful cue for ZSL, but an issue for GZSL, that has to tell them apart. In fact, our bias reduction boosts traffic signs and bicyclists so much that poles and unridden bicycles are not segmented anymore. Yet, unseen trucks are segmented without much altering seen vehicle segmentation, except for *other vehicles*.

It is remarkable that our framework proves to work both for classification and segmentation. Not all classification models adapt well to semantic segmentation. In fact, the two baselines DeViSe-3DSeg and ZSLPC-Seg originating from ZSL classification methods [18, 22] do not transfer well to semantic segmentation, despite our improvements.

4.5. Image-based 3D zero-shot learning

The auxiliary information for ZSL provides descriptions of unseen classes. Our method uses W2V and GloVe representations as auxiliary information, which are created from word co-occurrences in text corpora. We have shown in the previous sections that it is possible to meaningfully link these class prototypes to the point cloud representations.

We propose to investigate the use of an alternative auxiliary information based on image representations. As images capture the appearance of objects, visual representations should better link to object geometry. Here we describe an object class with a small set of images, and generate a visual representation by averaging features extracted using a CNN pre-trained on large datasets such as ImageNet [63]. Although rare, image-based representations have already been used for ZSL, e.g., human action recognition in videos [73].

We experiment with image representations as class prototype for 3D ZSL and GZSL. We consider 2 different image encoders, one pre-trained with supervision (ResNet-18 [30]) and one with self-supervision (ResNet-50 [12]). Experiments are run with the same parameters as for text embeddings. To generate prototypes for each class (seen and unseen), we average the features of the top-100 images obtained by a Google search with the class name. Results are shown in Table 3. Results for word embeddings are recalled from Tables 1 and 2, for comparison purposes.

	Classif.		*Segmentation*	
	ModelNet40		ScanNet	KITTI
Representation	ZSL	GZSL	HmIoU	
W2V+GloVe (self-sup.)	36.8	**41.3**	12.5	**17.1**
ResNet-18 [30] (sup.)	**43.6**	40.0	13.9	3.6
ResNet-50 [12] (self-sup.)	37.0	36.5	**15.5**	5.3

Table 3. Image-based 3D G/ZSL classification and segmentation.

For ZSL and GZSL, the two backbones performs as well or better than their counterpart for text embeddings, even without further tuning of the ϵ and β parameters (GZSL case). Image embeddings even outperform text embedding for segmentation on ScanNet, but fail on SemanticKITTI. A reason to this failure may be the presence of multiple classes per images (e.g. bicycle/ist, motorcycle/ist, sign/pole) leading to indistinguishable representations.

SemanticKITTI left aside, the good performance of the self-supervised representations underlines that this image-based approach can be a competitive alternative to the usual text-based ZSL. Indeed, in the same spirit as the non-supervised word embeddings of W2V, our image-based approach requires little or no supervision: merely the manual collection of a small quantity of images associated to given class names. Moreover, looking at the class-wise accuracy (for ResNet-18) on Fig. 4, we observe that the distribution of accuracies over the classes with an image-based representation is significantly different than with the text-based representations. In particular, classes like *nightstand* and *monitor* perform much better with image-based representations as auxiliary data. Taking the best of the two embeddings is a promising perspective and is left as future work.

5. Conclusion

In this study, we present a generative method for zero-shot learning on 3D point clouds. Experiments on a classification task shows that our method reaches the state of the art in both a classical and a generalized setting. Additionally, we show that our method can be easily extended to zero-shot semantic segmentation. To our knowledge, we are the first to tackle this task. We define natural baselines for 3D zero-shot segmentation, based on state-of-the-art approaches for classification, and compare them to our approach. Our method outperforms them on the three indoor and outdoor datasets we propose, based on S3DIS, ScanNet and SemanticKITTI. Besides, we introduce the use of image-based representations as an alternative auxiliary data for 3D ZSL and GZSL. We show that it is possible to outperform text-based representations in ZSL for classification. This experiments opens new perspectives as we observe that text embedding and image embeddings produce different performance distribution. Future work includes merging the two types of information to maximize zero-shot efficiency, as well as using phrasal (multi-word) embeddings to discover complex corner cases in large datasets.

Generative Zero-Shot Learning for Semantic Segmentation of 3D Point Clouds — Supplementary Material —

We present here complementary information on the 3DV 2021 paper "Generative Zero-Shot Learning for Semantic Segmentation of 3D Point Clouds". It provides more details and results on the following topics:

A. Generators:
   1. implementation,
   2. comparison,
   3. number of generated representations.

B. Classification:
   1. validation splits on ModelNet40,
   2. standard deviation, best and worst accuracy,
   3. sensitivity to the number of seen classes,
   4. ZSL confusion matrices,
   5. GZSL confusion matrices,
   6. classification results on McGill and SHREC2015.
   7. ablation study of bias reduction,

C. Semantic segmentation:
   1. cross-validation splits,
   2. semantic segmentation baselines,
   3. classwise performance,
   4. alternative splits,
   5. visualization of class prototype spaces,
   6. upper bound for semantic segmentation.

D. GZSL with image-based embeddings:
   1. image selection,
   2. feature averaging and normalization,
   3. classwise performance,
   4. visualization of class prototype spaces,
   5. sensitivity to the image collection quality.

After some cleaning, code should be available in the fall of 2021 on https://github.com/valeoai. Stay tuned.

A. Generators

We experimented with 4 different kinds of generators (see Section 4.2 of the paper):

- a denoising auto-encoder (DAE) [4],
- a generative moment-matching network (GMMN) [43],
- a conditional GAN (AC-GAN) [54],
- an adversial auto-encoder (AAE) [50].

We detail here their implementation and compare them.

A.1. Implementation details

To implement the generators, we basically follow the settings described in more details in [10, 11].

Each of these generators is made of 2 fully-connected layers, although with a different architecture.

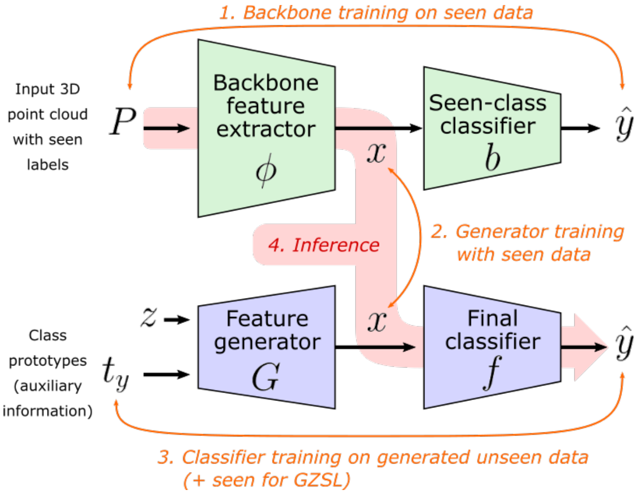

1. Backbone training on seen data
Input 3D point cloud with seen labels
P
Backbone feature extractor ϕ
x
Seen-class classifier b
$\hat{y}$
4. Inference
2. Generator training with seen data
z
Class prototypes (auxiliary information)
t_y
Feature generator G
x
Final classifier f
$\hat{y}$
3. Classifier training on generated unseen data (+ seen for GZSL)

Figure 2. (reminder) Training and inference: (1) backbone training on the seen classes, (2) generator training, (3) classifier training with artificial unseen features for ZSL (unseen and seen for GZSL), (4) inference through backbone and final classifier.

- The DAE consists of a encoder and a conditional decoder; it is trained with a mean square error loss.
- The AAE extends the DAE with a discriminator to constrain the latent code with an adversial criterion so that it follows a (normal) prior distribution; compared to the DAE, the AAE loss include an additional adversarial loss term.
- The AC-GAN generator is trained to produce conditional distributions similar to the true distributions; the training loss is the sum of a multi-label cross-entropy loss and an adversarial binary cross-entropy loss.
- Last, the GMMN is a conditional MLP trained with a loss penalizing the maximum mean discrepancy.

The number and size of the layers is kept consistent with [10, 11]; we did not try to optimize them. The experiments show that this setting also works well for generating 3D-like features from textual class prototypes as well as from 2D image features. More generally, we believe that generative ZSL methods developed for 2D images can be transferred well in this manner to handle 3D point clouds.

For training, we use the Adam optimizer [35] as in [10, 11], keeping the same parameters regarding the learning rate (decay) and the number of training epochs. As we use different datasets, the definition of one training epoch is however a bit different.

- For classification, we show every point cloud of the training set once in every epoch.

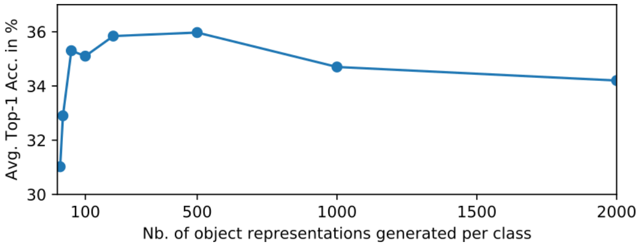


Figure 6. Cross-validation Acc (5-split average) wrt. the number of generated examples per class in ZSL classification (ModelNet40).

- For semantic segmentation, we use the default definition of an epoch for the respective backbone [5, 71].

A.2. Generator comparison

We ran cross-validation experiments with the four kinds of generators. We studied the behavior of the generators on ModelNet40 for classification and on S3DIS, ScanNet and SemanticKITTI for semantic segmentation.

For each dataset/task, we only used validation data created out of the available bases classes. The construction of validation sets is described in Section B.1 for classification, and Section C.1 for semantic segmentation. Results are presented in Table 4. We remark that the generators do not perform similarly and, more interestingly, that the best generator vary from one dataset/task to the other. This is an observation also made in [10].

A.3. Number of generated representations

With the cross-validation splits for ModelNet40, we study the impact of the number of generated examples for classification, as shown in Figure 6. The maximum top-1 accuracy is reached with 500 generated representations. We then use this number of generations for the (G)ZSL classification task.

Based on these experiments, we chose to use DAE for classification tests on ModelNet40 and GMMN for semantic segmentation tests on S3DIS, ScanNet and SemanticKITTI.

B. Classification

B.1. Validation splits on ModelNet40

Cross-validation is done with 5 splits on ModelNet40, i.e., on the 30 (seen) classes not in the 10 (unseen) classes of ModelNet10. Following [10], we select as validation classes in each split 20% of the seen classes, i.e., 6 classes.

B.2. Standard deviation, best and worst accuracy

The classification results presented in the paper for our method are averaged over 20 runs. We detail in Table 5 the standard deviations, as well as the results of the best and worst runs. We consider that the observed standard deviations, between 1.6 and 3.0 points of accuracy, are understandable and acceptable given the difference of modality between the text-based class prototypes and the 3D features, that have to be bridged. Interestingly, combining GloVe and W2V not only leads to a better accuracy but also to a lower standard deviation for the ZSL task. However, in the GZSL setting, the standard deviation of combining Glove+W2V is a bit higher compared to using only Glove or W2V only.

More detailed results are shown in Table 6, where we report the average accuracy and standard deviation for each class, over 20 runs. Following [18, 15, 16, 17], we report the global accuracy (also known as the Top-1 Accuracy): a prediction is considered as correct if it matches the ground-truth class. Additionally, we also report the class accuracy (Acc.), which is the (unweighted) average of the classwise accuracy over all classes. As the ModelNet40 test set is relatively balanced, the difference between the two kinds of average accuracies is small.

As noted in the paper, we observe that some classes like ‘*bed*’, ‘*desk*’, ‘*monitor*’ and ‘*nightstand*’ have a very low accuracy. Unsurprisingly, the standard deviation of the accuracy for these classes is low as well. More interestingly, the standard deviation tends to increase with the accuracy, except for class ‘*table*’, whose high accuracy is rather stable. It is difficult to know the reason of this behavior without more advanced studies, that would however be quite specific to this dataset, given its moderate size and variety with respect to all involved parameters. The causes of this high standard deviation may include: the absence of a similar-enough category among the seen classes; variations when training the generator, that never gets supervised information on unseen classes; weak correlations between textual embeddings and 3D features; and classification ambiguities. Yet, when considering all classes together, both the average class accuracy and the global accuracy show a moderate standard deviation (around 2 points of accuracy).

B.3. Sensitivity to the number of seen classes

To study the sensitivity of our method to the number of seen classes, we train our complete pipeline with a different number of seen classes on ModelNet40. Note that, for this analysis, we always evaluate the performance on the same 10 unseen classes of ModelNet40, but we train the networks using only on a subset of the 30 seen classes. We start from $N = 10$ and go on to $N = 30$ seen classes by selecting, each time, the first N classes in alphabetical order. For each N, we train all the networks used in our pipeline from scratch. We use the W2V class prototypes. For each N, we repeat the experiment 5 times and report the average scores obtained over these experiments.

We present in Figure 7 the global accuracy as a function of the number of seen classes. The list of selected seen

Dataset	Task	Setting	Metric	DAE	GMMN	AC-GAN	AAE
ModelNet40	classif.	ZSL	% HM	**36.0**	30.5	34.6	29.8
S3DIS	segment.	GZSL	% HmIoU	15.9	**18.6**	10.3	12.4
ScanNet	segment.	GZSL	% HmIoU	7.2	**9.1**	6.5	6.3
SemanticKITTI	segment.	GZSL	% HmIoU	4.4	**6.1**	4.1	4.3

Table 4. Comparison of the performance of the different generators for classification (on ModelNet40) and for semantic segmentation (on S3DIS, ScanNet and SemanticKITTI), based on the validation splits. The best generators regarding this validation data (DAE and GMMN) are kept for testing.

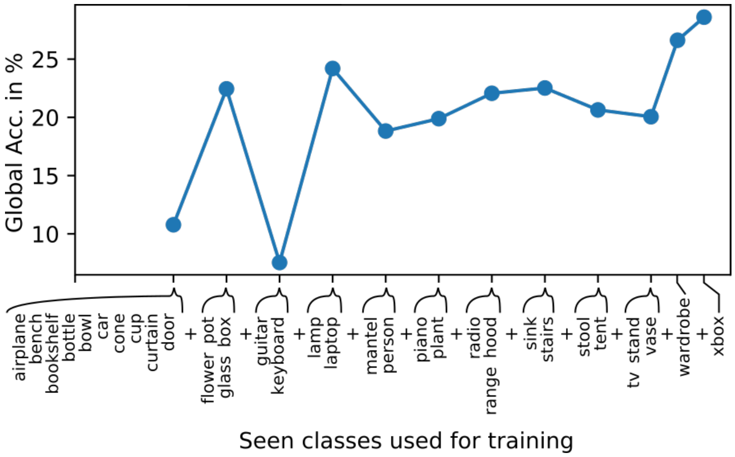


Figure 7. Impact of the number of seen classes for training on the ZSL classification task.

Setting	ZSL			GZSL		
Metric	Top-1 Acc (%)			Acc-HM (%)		
Word embedding	W2V	GloVe	GloVe +W2V	W2V	GloVe	GloVe +W2V
Worst	25.5	23.5	33.3	33.5	31.8	36.4
Average	28.6	29.3	36.8	36.6	34.7	41.3
Best	32.8	35.0	39.7	38.7	38.1	44.6
Std. deviation	2.1	3.0	1.7	1.6	1.9	2.1

Table 5. Variance study over 20 runs for zero-shot classification with our method on ModelNet40: ZSL Top-1 Acc (%) and GZSL seen-unseen Acc-HM (%), for different word embeddings.

classes is presented on the x-axis of the graph.

We notice that the results are unstable for a number of seen classes below 16 (‘*lamp*’, ‘*laptop*’) or below 18 (‘*mantel*’ and ‘*person*’). It is likely that the feature backbone is poorly trained, or that the generator is unable to “align” the space of class prototypes to the space of object representations when only a small number of seen classes is available. The performance stabilizes above 18 seen classes, with a noticeable jump in performance around 29 or 30 classes. A possible explanation for this jump could be that the class ‘*wardrobe*’ helps the alignment of the space of class prototypes onto the space of object representations because of some semantic closeness to the unseen classes ‘*dresser*’ and ‘*nightstand*’ and because of geometric similarities between point clouds belonging to these classes. To validate this kind of hypotheses, more similar experiments would be required, changing the last, 30-th class, or changing the position at which the class ‘*wardrobe*’ is included.

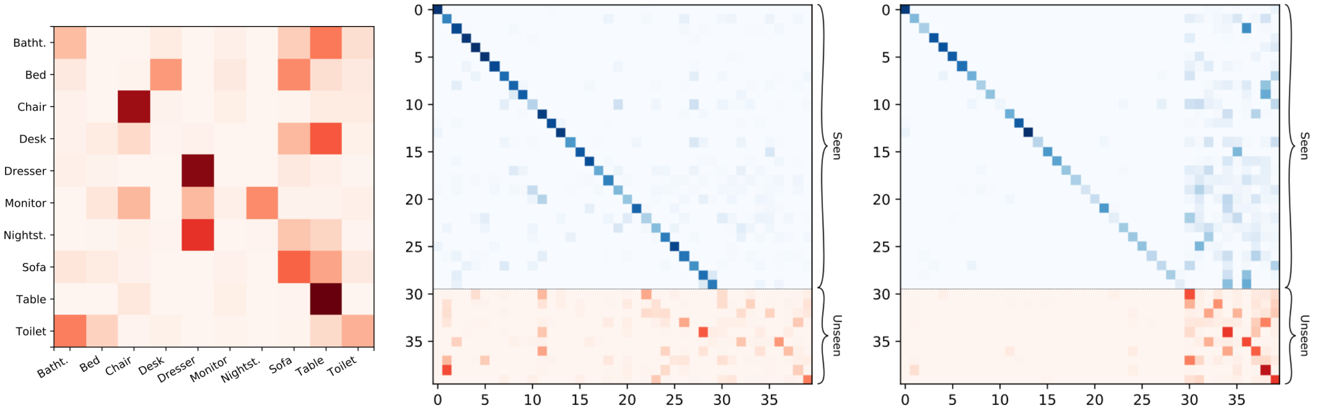
B.4. ZSL confusion matrices

As shown in Figure 8(a) for the ZSL setting, a lot of different classes are wrongly predicted as ‘*sofa*’ or ‘*table*’. It indicates that object representations generated for these two classes are similar to the representations of the classes that are falsely predicted as those two classes. In fact, these two classes are the equivalent of hubs as described in [15], except that we use here representations in the object domain.

The confusion matrix also shows that the class ‘*desk*’ is often mistaken for class ‘*table*’. The confusion in this case is probably caused by the fact that the two classes are close both semantically at text level and geometrically/visually at 3D appearance level. This also applies to class ‘*nightstand*’, that is often mistaken for class ‘*dresser*’. A close textual semantics (embeddings) for these two classes could lead to generate geometrical features that are similar, which would make it hard for the classifier to tell apart instances of the two classes. The fact is that classes ‘*nightstand*’ and ‘*dresser*’ look alike in the ModelNet40 dataset as the CAD

	Bathtub	Bed	Chair	Desk	Dresser	Monitor	Night stand	Sofa	Table	Toilet	Class Acc.	Global Acc.
W2V	45.0	3.8	37.7	2.2	54.0	2.0	0.1	9.8	94.8	40.7	29.0	28.6
std deviation	14.8	3.7	9.6	1.0	7.9	0.4	0.1	5.5	2.3	17.2	2.3	2.1
GloVe	21.6	0.0	81.7	3.8	37.9	1.8	1.1	65.1	16.6	53.6	28.3	29.3
std deviation	7.4	0.0	7.8	2.0	17.2	0.5	1.0	3.8	7.3	14.4	2.9	3.0
GloVe+W2V	22.3	0.1	81.3	1.1	85.2	3.0	0.1	47.5	91.5	25.5	35.8	36.8
std deviation	8.8	0.2	7.3	1.0	2.3	0.5	0.3	6.3	6.3	11.0	1.4	1.7

Table 6. Classwise classification accuracy in the classical ZSL setting for W2V, GloVe and GloVe+W2V on the ModelNet40 benchmark, i.e., using the 10 classes of ModelNet10 as unseen classes.



(a) ZSL

(b) GZSL with $\beta = 50, \epsilon = 0$

(c) GZSL with $\beta = 50, \epsilon = 0.995$

Figure 8. Confusion matrices for zero-shot classifications in ZSL and GZSL settings. The red color map is for unseen classes, and the blue color map for seen classes. Each row shows the distribution of the predictions of one class: the darker the color is, the more often object points of this class are predicted as the class of the column.

models are not in a metric scale.

We can note as well that nearly no object is ever predicted as '*monitor*', even actual monitors themselves. This could indicate that the generated object representation for the textual semantic embedding of this class is grossly wrong and does not come even close to the object representation of any other class. A reason for this behavior could be that the semantic embedding of '*monitor*' is an outlier in the class prototype space. Consequently, no knowledge from scene classes can help discovering this unseen class.

B.5. GZSL confusion matrices

In Figure 8(b) and Figure 8(c), we present the confusion matrices in the GZSL setting with $\beta = 50$ and either $\epsilon = 0$ or $\epsilon = 0.995$.

By comparing the two confusion matrices, we can observe the influence of the calibrated stacking in the resulting distribution. With $\epsilon = 0$, predictions are good for the seen classes, but unseen classes are rarely predicted. Using $\epsilon = 0.995$, which is the parameter maximizing the harmonic mean on the validation set, we clearly see a shift of the predictions toward the unseen classes, that in fact greatly improves accuracy. Seen classes are however negatively impacted with this shift as their prediction becomes less sharp; some of the seen classes that were correctly predicted with $\epsilon = 0$ may then be predicted as one of the unseen classes.

We can see the parameter ϵ as a way to counter the natural tendency of the network to classify all objects as belonging to a seen (supervised) class, and re-balance the predictions between seen and unseen classes.

B.6. Classification results on more datasets

Section 4.4.1 and Table 1 of the main paper compare our method to state-of-the-art ZSL and GZSL classification on the classic ModelNet40 [77] dataset. We complete here the comparison by experimenting as well on McGill [67] and SHREC2015 [45] datasets.

Following [18, 15, 16], we consider ModelNet40 classes as seen classes, and use unseen classes from McGill (14 classes, 115 examples) and SHREC2015 (30 classes, 192 examples). To be comparable with the literature, we follow [15] when adapting theses datasets for the ZSL and GZSL tasks. It means in particular that classes appearing in the seen classes of ModelNet40 are removed from the McGill and SHREC2015 dataset, and are not used as unseen classes. Besides, to be comparable with the experiments in

Dataset	Method	Gen-erative	ZSL			GZSL									
			W2V Acc.	GloVe Acc.	Glove + W2V Acc.	Bias reduction	W2V Acc. $\mathcal{S}$	W2V Acc. $\mathcal{U}$	W2V HM	GloVe Acc. $\mathcal{S}$	GloVe Acc. $\mathcal{U}$	GloVe HM	GloVe + W2V Acc. $\mathcal{S}$	GloVe + W2V Acc. $\mathcal{U}$	GloVe + W2V HM
ModelNet40	f-CLSWGAN* [82]	✓	20.7	-	-		76.3	3.7	7.0	-	-	-	-	-	-
	CADA-VAE* [65]	✓	23.0	-	-		**84.7**	1.3	2.6	-	-	-	-	-	-
	ZSLPC† [18]		28.0	20.9	20.5		40.1	22.5	28.8	49.2	18.2	26.6	-	-	-
	MHPC [15]		**33.9**	28.7	-	✓	53.8	26.2	35.2	**53.8**	25.7	**34.8**	-	-	-
	TZSLPC‡ [16]		23.5	-	-		83.7	0.4	0.8	-	-	-	-	-	-
	3DGenZ (ours)	✓	28.6	**29.3**	**36.8**	✓	48.8	**29.3**	**36.6**	44.7	**28.4**	34.7	47.8	36.5	41.3
McGill	f-CLSWGAN* [82]	✓	10.2	-	-		75.3	2.3	4.5	-	-	-	-	-	-
	CADA-VAE* [65]	✓	10.7	-	-		**83.3**	1.6	3.1	-	-	-	-	-	-
	ZSLPC† [18]		10.7	10.7	**16.1**		-	-	-	-	-	-	-	-	-
	MHPC [15]		12.5	**11.1**	-	✓	-	-	-	-	-	-	-	-	-
	TZSLPC‡ [16]		**13.0**	-	-		80.0	0.9	1.8	-	-	-	-	-	-
	3DGenZ (ours)	✓	8.4	7.2	9.4	✓	50.5	**7.2**	**12.5**	48.9	6.4	11.3	49.6	8.6	14.5
SHREC2015	f-CLSWGAN* [82]	✓	5.2	-	-		74.2	0.8	1.6	-	-	-	-	-	-
	CADA-VAE* [65]	✓	**6.2**	-	-	-	80.0	1.7	3.3	-	-	-	-	-	-
	ZSLPC† [18]		5.2	3.6	**6.8**		-	-	-	-	-	-	-	-	-
	MHPC [15]		**6.2**	**4.1**	-	✓	-	-	-	-	-	-	-	-	-
	TZSLPC‡ [16]		5.2	-	-		**82.1**	0.9	1.8	-	-	-	-	-	-
	3DGenZ (ours)	✓	4.9	**4.1**	4.9	✓	54.1	**4.3**	**8.0**	47.2	3.9	7.2	50.9	4.6	8.4

Table 7. ZSL and GZSL classification results (in %) on ModelNet40 [77], McGill [67] and SHREC2015 [45]. For a fair comparison, we report results based on the same PointNet backbone. Results are averaged over 20 runs for 3DGenZ (not other methods). Missing figures are due to code unavailable and previous publications not evaluating with all kinds of word embeddings.

*: adaptation of 2D methods to 3D point clouds, implemented in [16].

†: best reported variant in [18], i.e., PointNet + NetVlad.

‡: inductive baseline reported in [16].

[18, 15, 16], we use the same PointNet backbone as we already used for the experiments with the ModelNet40 unseen classes. Furthermore, we use the same hyperparameters for the bias reduction as we used for the ModelNet40 unseen classes, as the same seen classes are used and no additional cross-validation is necessary.

Results are shown in Table 7. (For a complete overview, we also recall in Table 7 the results on ModelNet40, that were already provided in Table 1 of the main paper.) Please note that missing figures are due to code unavailable and to previous publications not evaluating with all kinds of word embeddings.

These test datasets are difficult challenges for ZSL and GZSL as the 30 classes of SHREC2015 as well as 10 of the 14 classes in the McGill datasets are animals, whereas the ModelNet40 seen classes used for training do not contain a single animal and focus on man-made objects. This is probably a reason why, in Figure 3 of [18], the t-SNE visualisation of the McGill and ModelNet40 word representations looks quite disjoint. Morevoer, it can be noted that the number of test examples for SHREC2015 and McGill is quite low compared to the 908 test examples for unseen classes in ModelNet40. More variation from one method to another can thus be expected with these two datasets.

Indeed, compared to the results with ModelNet40 unseen classes, an overall drop in Top-1 Acc. and HM can be observed on McGill and SHREC2015 unseen classes, both in our results and with the other methods. It corroborates the difficulty of the task mentioned above.

For the ZSL task, we are a bit below the state of the art with word2vec (W2V) embeddings, but we are comparable with Glove embeddings (better on ModelNet40, not as good on McGill, equal on SHREC2015). For the GZSL task, which is also relevant for semantic segmentation, we see that our framework with bias reduction achieves state-of-the-art results (HM) on all three datasets with W2V embeddings, and is comparable to the best method using GloVe embeddings (on ModelNet40, which is the only dataset tested in the literature with these embeddings).

(In a recent unpublished work [17], better results are obtained on McGill with a different backbone network. However, as the choice of the backbone network, for the same ZSL or GZSL method, has a large impact on performance, as can be seen in Tables 2 and 3 of [17], these results are not directly comparable to those reported in Table 7 nor in the main paper.)

These ZSL and GZSL classification results validate our approach and suggest it is relevant to use it as a general framework to also derive a semantic segmentation method.

B.7. Ablation study of bias reduction

Section 4.3 and Figure 3 of the main paper provide an ablation and a parameter range study for ZSL classification (on ModelNet40), and for GZSL semantic segmentation (on S3DIS, ScanNet and SemanticKITTI). We complete here this ablation study with the case of GZSL classification.

Table 8 reports the results for GZSL classification on ModelNet40 for our 3DGenZ framework with and without our bias reduction method. It can be seen clearly that our bias reduction improves the HM and the accuracy of unseen classes for all kinds of word embeddings. Because bias reduction provides a trade-off, this improvement comes with a drop of the performance in the seen classes, as already described in Section B.5.

C. Semantic segmentation

C.1. Cross-validation splits

Cross-validation is done with 5 splits on S3DIS, 4 splits on ScanNet and 3 splits on SemanticKITTI. Following [10], we select 20% of the seen classes in each split as validation classes with a minimum of at least 2 classes. Therefore, we have 2, 3 and 3 selected validation classes in S3DIS, ScanNet and SemanticKITTI, respectively.

For each validation split, as we are in the inductive zero-shot setting, the feature backbone is trained using only seen-class data that do not include classes selected as validation classes, and the validation evaluations are done only on the validation classes. We would also like to highlight again that the unseen classes for testing are *not* used in the validation process.

As explained in the main part of the paper, frequently appearing classes in the semantic segmentation datasets cannot be used as validation classes because removing them would drastically reduce the size of the training dataset. We present in Table 9 our choices of validation classes. These chosen validation classes avoid reducing too much the amount of remaining training data.

C.2. Baselines for semantic segmentation

To our knowledge, this work is the first[1] to address zero-shot semantic segmentation for point clouds. To show the efficiency of our proposed approach, we designed two baselines based on previous work for zero-shot classification.

As baselines, we adapted the ZSLPC [18] and DeViSE [11] methods to semantic segmentation. However, a direct adaption did not produce valuable results as we observed a strong prediction bias towards the seen classes on all datasets we experimented with (see lines for ZSLPC-Seg* and DeViSe-3DSeg* in Table 2 of the paper). A bias reduction mechanism was thus also needed for these two baseline methods.

However, these methods use a different paradigm than ours: they base classification on a nearest-neighbor search in the space of class prototypes, whereas we produce classification scores and pseudo-probabilities (after softmax) via a trained classifier. As a result, it was not possible to apply the same bias reduction techniques that we used (class-dependent weighting and calibrated stacking). Nevertheless, we tried to rebalance unseen classes by reducing the distance to prototypes of unseen classes by a constant value; it is somehow similar to the calibrated stacking we are using, where the pseudo-probability of unseen classes is increased by ϵ, but it is in a totally different and much larger space. Unfortunately, it did not lead to valuable results.

To construct meaningful baselines, we thus had to design a more complex bias reduction technique. To reduce the bias towards seen classes, we proceed as follows: we search the K-nearest-neighbors in the space of class prototypes; if a prototype of an unseen class is present among these K neighbors, we pick the class of the nearest prototype of unseen class; otherwise, we pick the class of the nearest prototype. We call the corresponding methods ZSLPC-Seg and DeViSe-3DSeg, respectively.

As described in the paper, we select the best performing value for K depending on the dataset (see Table 11).

C.3. Classwise performance on segmentation

C.3.1 Performance details on S3DIS

The GZSL semantic segmentation method presented in this paper achieves a HmIoU of 12.9% on the test data of S3DIS (Table 1 of the main paper). The detailed classwise results are shown here in Table 10.

According to per-class mIoU, the best performing unseen class is '*Beam*', and the worst is '*Column*'. However, a relatively bad performance of the class '*Column*' can also be seen in the full-supervised learning scenario (FSL). This could indicate that this class is based on visual features that are hard to differentiate from other classes.

Regarding classwise accuracy, '*Sofa*' also performs very well. Yet, a large Acc with a low mIoU is the indication that points that should be labeled with another class (than '*Sofa*') are actually mispredicted as '*Sofa*'. The per-row normalized confusion matrix in Figure 9 supports this assumption, as 35% of the '*Chairs*' are predicted as '*Sofas*'. A reason could be that while substantially different geomet-

[1] An unpublished report on this topic was recently made public [46]. However, it operates in a different setting as the unseen classes are present at training time, although unlabeled. In our inductive setting, strictly no unseen class can seen at training time, which makes the task substantially more difficult and which also reduces the size of training data to satisfy this constraint.

Method	GZSL									
	Bias reduction	W2V			GloVe			GloVe + W2V		
		Acc. $\mathcal{S}$	Acc. $\mathcal{U}$	HM	Acc. $\mathcal{S}$	Acc. $\mathcal{U}$	HM	Acc. $\mathcal{S}$	Acc. $\mathcal{U}$	HM
3DGenZ w/o bias reduct.		**79.3**	9.96	17.6	**79.7**	12.1	21.0	**79.0**	13.4	22.8
3DGenZ w/ our bias reduct.	✓	48.8	**29.3**	**36.6**	44.7	**28.4**	**34.7**	47.8	**36.5**	**41.3**

Table 8. Ablation study: GZSL classification with 3DGenZ on ModelNet40 with and without our bias reduction mechanism.

Datasets	Split 1	Split 2	Split 3	Split 4	Split 5
S3DIS	Door, Bookcase	Bookcase, Board	Door, Table	Table, Chair	Chair, Board
ScanNet	Chair, Table, Cabinet	Counter, Bathtub, Sink	Counter, Table, Sink	Chair, Cabinet, Bathtub	-
SemanticKITTI	Other-vehicle, Person, Motorcyclist	Bicycle, Person, Other-ground	Other-vehicle, Motorcyclist, Other-ground	-	-

Table 9. Classes used for validation in the different cross-validation splits for the semantic segmentation task.

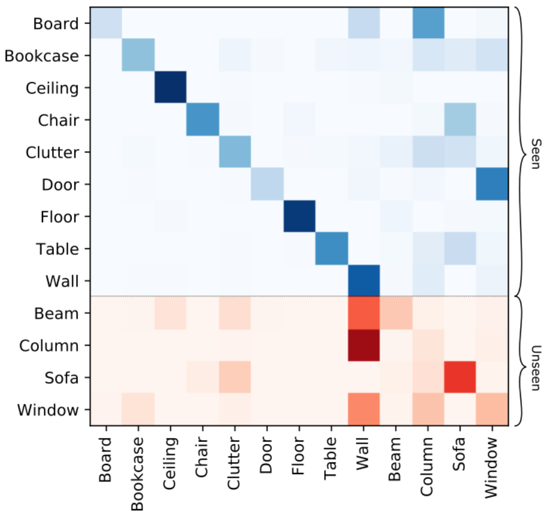


Figure 9. Confusion matrix for the GZSL semantic segmentation of S3DIS with 3DGenZ. The red color map is for unseen classes, and the blue one for seen classes. Each row shows the distribution of predictions of one class: the darker the color is, the more often points of this class are predicted as the class of the column.

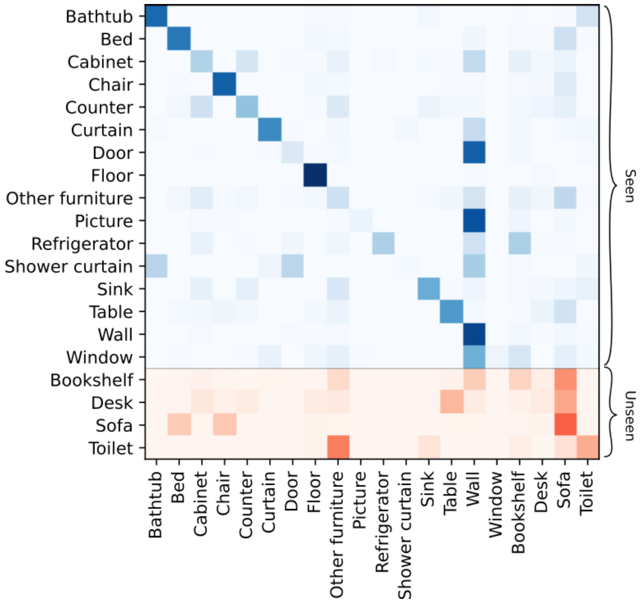


Figure 10. Confusion matrix for the GZSL semantic segmentation of ScanNet with 3DGenZ. The red color map is for unseen classes, and the blue one for seen classes. Each row shows the distribution of predictions of one class: the darker the color is, the more often points of this class are predicted as the class of the column.

rically, these two classes are close regarding textual semantic, hence word embeddings.

It can also be seen in the confusion matrix that the classes '*Beam*', '*Column*' and '*Window*' are often falsely predicted as '*Wall*', which could be due to the fact that all these classes similarly feature large flat smooth surfaces.

The confusion matrix additionally shows that, probably due to remaining weight issues, some seen classes have a tendency to be predicted with the label of an unseen class, although for these seen classes the classifier is presented with the actual 3D features, as opposed to generated ones. For example, the class '*Board*' is falsely predicted as '*Column*' in 54% of the cases. This effect probably also contributes to many '*Chair*' points being predicted as '*Sofa*'.

S3DIS split		seen classes									**unseen classes**				
		Board	Bookcase	Ceiling	Chair	Clutter	Door	Floor	Table	Wall	**Beam**	**Column**	**Sofa**	**Window**	Hm IoU
FSL	mIoU	53.9	54.4	96.5	75.9	66.0	78.7	96.0	70.3	74.1	63.1	10.2	54.1	72.4	59.6
3DGenZ	mIoU	19.1	34.1	92.8	56.3	39.2	25.4	91.5	57.3	62.3	13.9	2.4	4.9	8.1	12.9
3DGenZ	Acc.	19.7	39.8	96.9	58.7	43.5	25.8	92.9	61.9	80.3	20.0	9.1	62.4	23.7	-

Table 10. Classwise GZSL semantic segmentation performance (%) on the S3DIS split: fully-supervised learning (FSL), i.e., training using annotations for both seen and unseen classes, as upper bound, and GZSL with 3DGenZ with respect to unseen classes (in bold face).

Method \ Dataset	S3DIS	ScanNet	SemanticKITTI
ZSLPC-Seg	5	2	5
DeViSe-3DSeg	7	2	5

Table 11. Best value of bias reduction parameter K for the baseline GZSL semantic segmentation methods.

C.3.2 Performance details on ScanNet

To complete the results presented in Table 1 of the paper, the classwise performance in the FSL and GZSL settings on the test set of ScanNet are reported in Table 12. The confusion matrix for the GZSL scenario is shown in Figure 10.

Like for S3DIS, a number of seen classes are wrongly classified as '*Wall*'. Again, it could be due to large flat surfaces that are shared by all of the misclassified classes.

In the unseen classes, an intriguing observation is the bias of the '*Desk*' and '*Bookshelf*' classes towards the '*Sofa*' class. We hypothesize that features extracted by the backbone for examples of '*Bookshelf*' and '*Desk*' are closer to generated representations for '*Sofa*' than to generated representations of their own class. As several seen classes are also wrongly classified as '*Sofa*', we suspect that the '*Sofa*' class accumulates hard-to-classify examples besides examples of its own class, in a hub-like effect [15] already observed for classification (see Section B.4).

The unseen class '*Desk*' is also often misclassified as the seen class '*Table*'. This ambiguity exists both on the geometrical and on the semantic level.

Another observation is that, compared to the SemanticKITTI and the ModelNet40 datasets, the bias correction is less strong towards unseen classes.

C.3.3 Performance details on SemanticKITTI

Table 13(a) provides the classwise semantic segmentation performance for our method on the SemanticKITTI dataset (main split). As for S3DIS and ScanNet, the Acc is much larger than the mIoU for unseen classes. The confusion matrix in Figure 11 confirms here as well that it is due to points of seen classes being predicted as some unseen class. For example, the instances of the seen class '*Other vehicle*' are predicted in 91% of the cases as the unseen class '*Truck*', and the seen class '*Person*' is predicted in 78% of the cases as the unseen class '*Motorcycle*'.

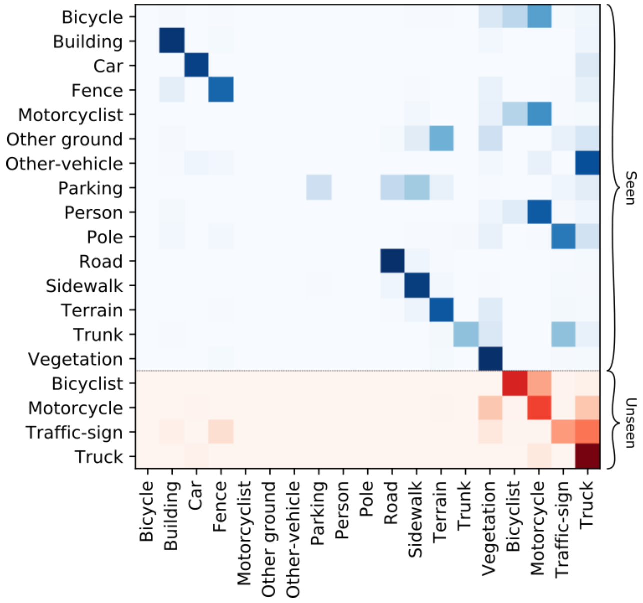


Figure 11. Confusion matrix for the GZSL semantic segmentation of SemanticKITTI with 3DGenZ. The red color map is for unseen classes, the blue one for seen classes. Each row shows the distribution of predictions of one class: the darker the color is, the more often points of this class are predicted as the class of the column.

There is also a cluster of classes whose textual semantics and 3D appearance are strongly connected, which might cause some confusion. These are the classes '*Bicycle*', '*Motorcycle*', '*Bicyclist*', '*Motorcyclist*' and '*Person*'. Classes '*Motorcyclist*' and '*Bicyclist*' are used for the person as well as the motorcycle if this person is on a (motor)bike. The prediction on the seen classes '*Bicycle*', '*Person*' and '*Motorcyclist*' is in the majority of the cases distributed between the unseen classes of '*Motorcycle*' and '*Bicyclist*'. For the mentioned seen classes, it is very harmful; in fact, they have a mIoU of 0%.

It illustrates that, in this kind of setting, the class-dependent weighting and the calibrated stacking may turn the bias towards seen classes into a bias towards unseen

ScanNet split	seen classes																unseen classes				
	Bathtub	Bed	Cabinet	Chair	Counter	Curtain	Door	Floor	Other furniture	Picture	Refrigerator	Shower curtain	Sink	Table	Wall	Window	**Bookshelf**	**Desk**	**Sofa**	**Toilet**	Hm IoU
FSL mIoU	58.0	67.5	21.2	75.5	12.0	35.2	13.6	96.5	20.6	10.7	39.9	63.3	34.2	59.5	81.1	4.8	56.9	30.0	57.4	63.4	47.2
3DGenZ mIoU	64.9	44.0	16.9	63.2	15.3	33.8	10.4	91.0	10.1	4.3	26.1	0.2	27.5	43.1	71.3	2.8	6.3	3.3	13.1	8.1	12.5
3DGenZ Acc.	75.7	68.3	27.6	78.1	35.4	40.2	12.1	97.4	18.5	5.1	31.3	0.3	44.3	56.0	83.2	3.1	13.4	5.9	49.6	26.3	-

Table 12. Classwise GZSL semantic segmentation performance (%) on the ScanNet split: fully-supervised learning (FSL), i.e., training using annotations for both seen and unseen classes, as upper bound, and GZSL with 3DGenZ with respect to unseen classes (in bold face).

(a) Semantic KITTI split 1 (main)	seen classes															unseen classes				
	Bicycle	Building	Car	Fence	Motorcyclist	Other ground	Other vehicle	Parking	Person	Pole	Road	Sidewalk	Terrain	Trunk	Vegetation	**Bicyclist**	**Motorcycle**	**Traffic sign**	**Truck**	Hm IoU
FSL mIoU	42.0	88.6	93.6	65.8	0.0	2.7	41.1	28.9	69.7	63.7	89.4	77.1	70.5	70.7	87.5	74.4	58.6	26.7	41.6	54.5
3DGenZ mIoU	0.0	87.3	86.9	61.8	0.0	0.0	0.0	18.6	0.0	0.0	88.8	78.6	73.6	38.2	87.8	28.0	11.5	0.9	2.6	17.1
3DGenZ Acc.	0.0	91.5	87.4	74.8	0.0	0.0	0.0	19.9	0.0	0.0	93.3	89.2	79.8	38.6	94.0	66.8	57.2	32.7	90.8	-

(b) Semantic KITTI split 2 (altern.)	seen classes															unseen classes				
	Bicyclist	Building	Car	Fence	Motorcycle	Other ground	Other vehicle	Parking	Person	Pole	Road	Sidewalk	Terrain	Trunk	Vegetation	**Bicycle**	**Motorcyclist**	**Traffic sign**	**Truck**	Hm IoU
FSL mIoU	74.4	88.6	93.6	65.8	58.6	2.7	41.1	28.9	69.7	63.7	89.4	77.1	70.5	70.7	87.5	42.0	0.0	26.7	41.6	38.8
3DGenZ mIoU	0.0	84.5	78.9	53.5	3.9	0.0	0.0	21.8	0.0	0.0	85.4	72.6	67.8	50.1	87.9	0.0	0.3	3.0	2.0	12.7
3DGenZ Acc.	0.0	91.6	80.1	73.3	14.9	0.0	0.0	0.0	0.0	91.7	87.3	25.6	75.3	55.1	95.4	0.0	51.1	25.6	30.8	-

(c) Semantic KITTI split 3 (altern.)	seen classes															unseen classes				
	Building	Car	Fence	Other ground	Other vehicle	Parking	Person	Pole	Road	Sidewalk	Terrain	Traffic sign	Truck	Trunk	Vegetation	**Motorcycle**	**Motorcyclist**	**Bicyclist**	**Bicycle**	Hm IoU
FSL mIoU	88.6	93.6	65.8	2.7	41.1	28.9	69.7	63.7	89.4	77.1	70.5	26.7	41.6	70.7	87.5	58.6	0.0	74.4	42.0	51.0
3DGenZ mIoU	82.4	82.8	47.2	0.0	0.0	15.3	0.0	0.0	82.9	70.2	0.0	66.9	0.0	0.1	88.5	0.9	1.9	0.1	0.0	1.4
3DGenZ Acc.	94.4	82.9	57.7	0.0	0.0	17.6	0.0	0.0	92.1	82.3	0.0	77.1	0.0	0.1	96.0	18.3	74.5	3.0	0.0	-

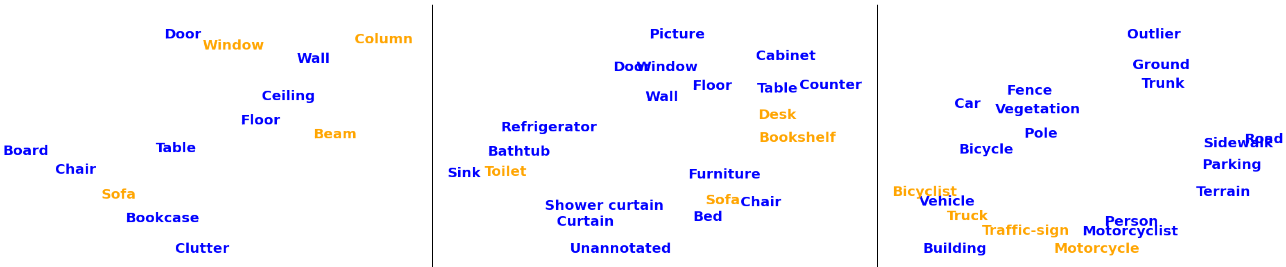
Table 13. Classwise semantic segmentation performance (%) on SemanticKITTI using main split 1 (a) or alternative splits 2 (b) and 3 (c): fully-supervised learning (FSL), i.e., training using annotations for both seen and unseen classes, as upper bound performance, and GZSL with 3DGenZ w.r.t. unseen classes.

classes, although their parameters β and ϵ are specifically and systematically adapted to the dataset. However, these bias-reduction parameters are set by cross-validation using validation-unseen classes that are unrelated test-unseen classes, which could be an issue.

C.4. Alternative splits

As the 3D ZSL segmentation task is new, no benchmark is available to evaluate our method. We thus had to make our own benchmarks, creating class splits in existing 3D semantic segmentation datasets and curating data for inductive ZSL (strictly no unseen class in training data).

To create these splits, as already stated, one of the concerns is to keep as much training data as possible, which favors less represented classes as unseen classes. However, the choice of unseen classes also defines the difficulty of the benchmarks. Section 4.1 of the paper presents the rationale of the split choices for S3DIS, ScanNet and SemanticKITTI. As we hope that our benchmarks will be used for further research, we define here two additional splits on SemanticKITTI, that we see as even more challenging than the split we present in the paper, and we evaluate our



(a) S3DIS (b) ScanNet (c) SemanticKITTI (split 1 for color)

Figure 12. t-SNE [72] visualizations of the W2V+GloVe class prototypes for semantic segmentation datasets (**blue seen**, **orange unseen**).

method on them.

In the alternative split 2 for SemanticKITTI, we start from the split presented in the paper (main split 1) and we replace the role of bicycles and motorbikes, i.e., '*motorbikes*' and '*bicyclists*' are now seen, while '*motorcyclists*' and '*bicycles*' are now unseen. The motivation for this split is exactly the same as the one we describe in the paper, i.e., allowing to leverage on closely related classes. However, we see this other split as more challenging, at least for our backbone, as the mIoU for '*bicycle*' is comparatively lower in the FSL scenario; furthermore, the mIoU is even 0.0% for '*motorcyclist*'. Results for this alternative split are reported in Table 13(b). Our model achieves on this split an HmIoU that is 4.4 points lower than the HmIoU of 17.1% achieved on the split used in the paper (split 1). We assume it is linked to the intrinsic difficulty of classifying '*bicycle*' and '*motorcyclist*', as highlighted in the FSL scenario.

An even more difficult scenario is the selection of classes '*Bicycle*', "*Bicyclist*', '*Motorcycle*' and '*Motorcyclist*' as unseen, given that these four classes are all semantically and geometrically very close and that it is difficult to tell them apart. This assumption is confirmed by the results on this alternative split 3, reported in Table 13(c). We achieve an HmIoU of only 1.4%. While the accuracy of '*Motorcyclist*' is quite high, the very low mIoU shows that it is very hard to distinguish the different unseen classes and a lot of examples are wrongly classified as '*Motorcyclist*'.

C.5. Visualisation of class prototype spaces

To assess the difficulty of transferring knowledge of seen classes to unseen classes, we examine the textual semantic similarities and dissimilarities of word embeddings. To that end, we provide in Figure 12 t-SNE visualizations [72] of the W2V+GloVe class prototypes for the semantic segmentation datasets (S3DIS, ScanNet, SemanticKITTI). These diagrams can be compared to the respective performance Tables 10, 12, 13 and confusions matrices in Figures 9-11.

C.6. Upper bound for semantic segmentation

Following the idea of a helpful anonymous reviewer, we experimented with 3D features as class prototypes, only to get a kind of upper bound as this does not satisfy the zero-shot principle. We trained the 3D backbone under full supervision using only seen classes. Then we created class prototypes using the ground-truth segmentation masks of seen and unseen classes, averaging features of all correctly-classified points for each seen class, and features of all points for each unseen class. Finally, we used our method with these "ideal" prototypes instead of the word- or image-based prototypes. Because these prototypes are obtained using knowledge about the 3D backbone, we had to reduce our bias term to zero. The results are summarized in the Table 14. They suggest that we are probably close to the best results one can hope for with this kind of generative approach with that backbones.

	ZSL			Supervised	
HM	W2V& Glove	image	"ideal"	ZSL backb.	Full superv.
S3DIS	12.9	5.7	21.0	31.8	59.6
ScanNet	12.5	15.5	17.0	40.3	47.2
Sem.KITTI	17.1	5.3	17.5	21.2	54.5

Table 14. Comparing the ZSL results with the upper bound ("ideal") and with supervised models.

D. Image-based representations

To built our class prototypes based on image representations, we use images of objects belonging to each class of the datasets and extract deep features for each of these images using a pre-trained CNN on ImageNet [63]. Details about the selection of the images are given in Section D.1 and feature extraction is described in Section D.2. In Section D.3 the classwise results for the different datasets are given and in Section D.5 the sensitivity to the image collection quality is discussed.

D.1. Image selection

For each of the classes, we collect the first 100 images returned by a Google image search with the corresponding class name, using the option to select images with a major-

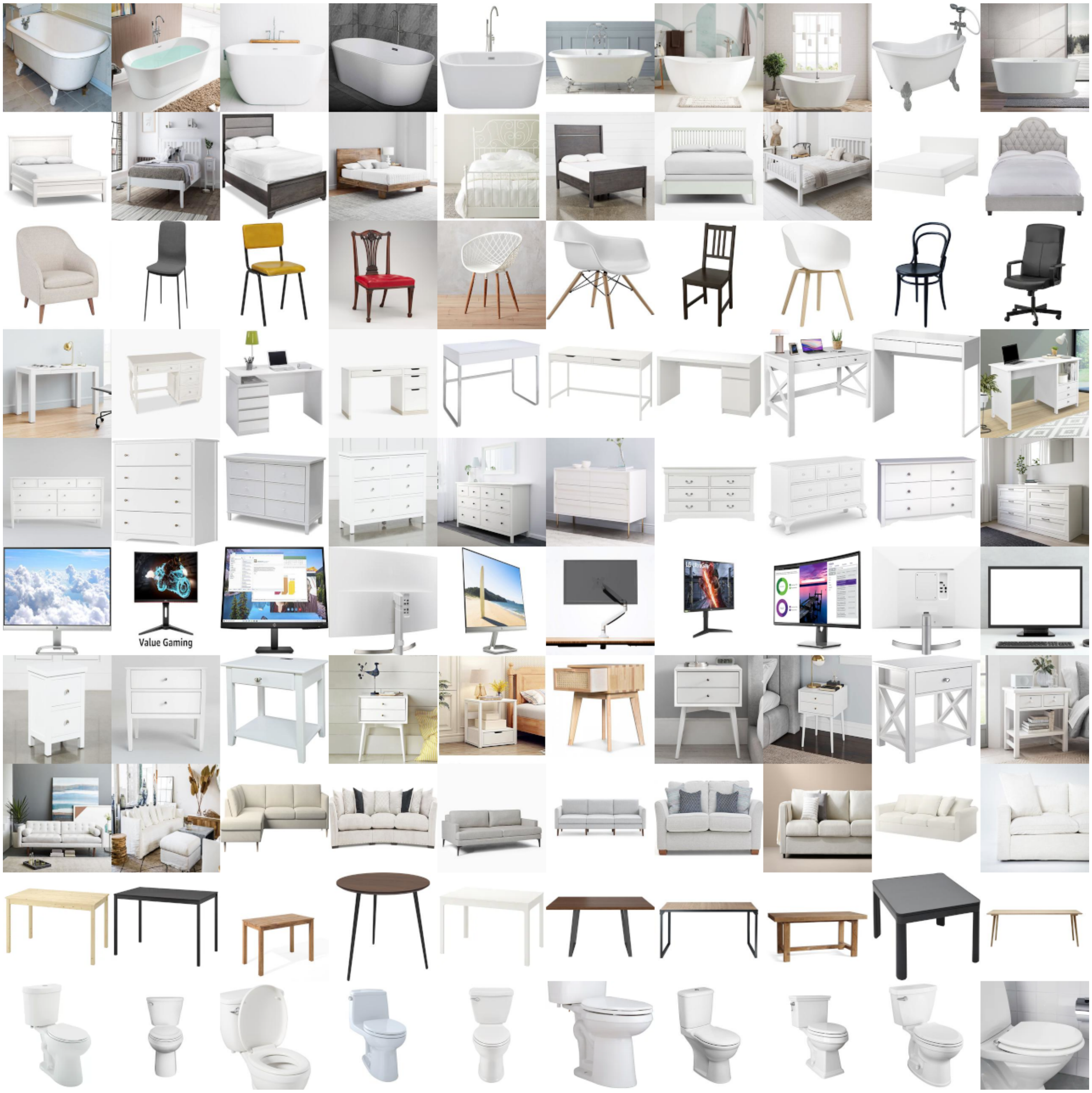


Figure 13. The first ten images collected with Google Images and used to generate the image embeddings of ModelNet40 (unseen classes).

ity of white pixels. This search setting is used to favor the selection of images containing only one object, typically on a white background. We show in Figure 13 the first ten images obtained for the unseen classes of ModelNet40 using this procedure. It can be seen that most of these images indeed contain only one object of the desired class.

Our reason for such a setting is that we would like the CNN to extract features that are specific to each object class, and also with less noise coming from background pixels. Note that this use of images in the wild comes with strictly no annotation effort, in the spirit of zero-shot learning. The paper actually reports results with a network pre-trained with self-supervision [29], as well as with a network pre-trained with full supervision.

Please also note that, although ImageNet features a thousand classes, a number of seen and unseen classes in our datasets are not classified in ImageNet (which however does not mean they cannot appear in the background), e.g.,

(a) S3DIS	supervis.	seen classes									**unseen classes**				
		Board	Bookcase	Ceiling	Chair	Clutter	Door	Floor	Table	Wall	**Beam**	**Column**	**Sofa**	**Window**	Hm IoU
W2V+Glove	self	19.1	34.1	92.8	56.3	39.2	25.4	91.5	57.3	62.3	13.9	2.4	4.9	8.1	**12.9**
ResNet-18 [30]	full	43.3	36.5	92.7	67.7	33.8	45.9	90.8	62.9	65.5	0.8	0.7	4.9	5.5	5.7
ResNet-50 [12]	self	25.0	37.0	93.5	64.1	36.8	34.6	91.4	55.4	65.4	0.3	1.0	3.4	3.9	4.1

(b) ScanNet	supervis.	seen classes																**unseen classes**				
		Bathtub	Bed	Cabinet	Chair	Counter	Curtain	Door	Floor	Other furniture	Picture	Refrigerator	Shower curtain	Sink	Table	Wall	Window	**Bookshelf**	**Desk**	**Sofa**	**Toilet**	Hm IoU
W2V+Glove	self	64.9	44.0	16.9	63.2	15.3	33.8	10.4	91.0	10.1	4.3	26.1	0.2	27.5	43.1	71.3	2.8	6.3	3.3	13.1	8.1	12.5
ResNet-18 [30]	full	50.2	40.6	15.7	61.1	8.3	32.9	9.7	90.9	5.2	0.8	26.3	0.1	24.7	43.8	72.1	3.8	8.9	15.6	8.0	3.7	13.9
ResNet-50 [12]	self	56.7	42.7	16.0	59.4	13.0	35.0	10.8	90.9	7.2	0.4	29.7	11.9	25.6	40.2	72.4	3.4	10.7	15.7	11.3	3.1	**15.5**

(c) Semantic KITTI (main split, 1)	supervis.	seen classes															**unseen classes**				
		Bicycle	Building	Car	Fence	Motorcyclist	Other ground	Other vehicle	Parking	Person	Pole	Road	Sidewalk	Terrain	Trunk	Vegetation	**Bicyclist**	**Motorcycle**	**Traffic sign**	**Truck**	Hm IoU
W2V+Glove	self	0.0	87.3	86.9	61.8	0.0	0.0	0.0	18.6	0.0	0.0	88.8	78.6	73.6	38.2	87.8	28.0	11.5	0.9	2.6	**17.1**
ResNet-18 [30]	full	0.0	85.6	93.3	66.0	0.0	0.0	0.0	0.3	0.0	0.0	87.7	75.3	70.3	62.4	87.4	1.5	0.3	0.0	5.7	3.6
ResNet-50 [12]	self	0.0	86.4	93.0	61.4	0.0	0.0	0.8	7.2	0.0	0.0	88.9	77.9	72.8	62.3	88.0	4.1	1.9	0.0	5.3	5.3

Table 15. Classwise semantic segmentation performance (IoU in %) on datasets S3DIS (a), ScanNet (b) and SemanticKITTI (c), with three different kinds of embeddings as class prototypes: (1) W2V+GloVe word embeddings, (2) image embeddings from a ResNet-18 fully supervised on ImageNet [30], (3) image embeddings from a ResNet-50 self-supervised on ImageNet [12]. Unseen classes are in bold face.

seen '*Ceiling*' and '*Floor*', and unseen '*Column*' in S3DIS; seen '*Counter*' and '*Sink*' in ScanNet; seen '*Building*' and '*Road*', and unseen '*Bicyclist*' in SemanticKITTI.

In any case, even objects of classes that appear both among ImageNet categories and among the classes of our 3D datasets come with very different modalities, i.e., image vs point cloud. Besides, they are never directly associated as we only use the image-based pre-trained network to create embeddings from the images selected as described above.

D.2. Construction of image-based representations

For each of the 100 images that we collected for each class, as described in Section D.1, we extract the image features obtained after the global average pooling layer of the pre-trained CNN. These 100 image features are then averaged for each class before being ℓ_2-normalised. This constructs the image-based class prototypes that we used for the ZSL and GZSL tasks (Section 4.5 and Table 3 of the paper).

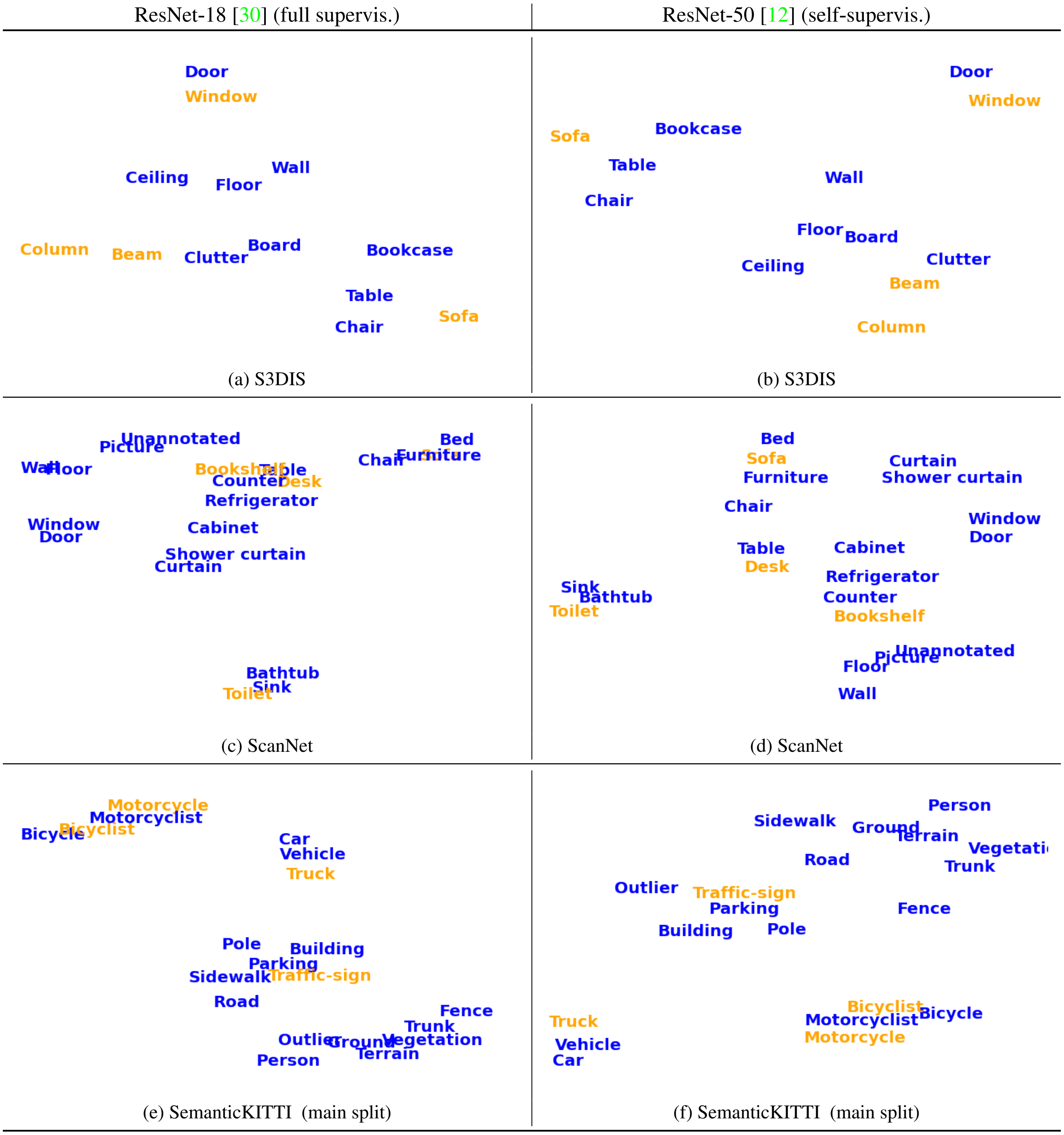
D.3. Classwise results for semantic segmentation

We report in Table 15 our classwise GZSL performance on datasets S3DIS, ScanNet and SemanticKITTI, when using either word embeddings or image embeddings. The word embeddings are the concatenation W2V+GloVe. The image-based representations are extracted using a ResNet-18 trained under full supervision on ImageNet [30], or a ResNet-50 trained by self-supervision on ImageNet [12].

With S3DIS, in Table 15(a), we observe relatively similar results on seen classes, whether we use word or image embeddings. However, regarding unseen classes, we get a significantly lower IoU for classes '*Beam*' and '*Column*' using the image-based representation, compared to the word-based representation. We suppose that the quality of the retrieved images for these classes, mainly due to ambiguities, explains the poor performance. As a matter of fact, '*Beam*' images are disparate, containing, e.g., images of light beam or of the drink Jim Beam, and many of the '*Column*' images picture antique columns, or columns from Excel sheets..

Likewise, the results reported in Table 15(b) on ScanNet show that the performance is nearly independent of the type of class prototype on the seen classes. A possible explanation for the drop of performance on the class '*Picture*', in particular with the image-based representations, could be a confusion between the scene that is pictured and the picture itself. We hypothesize that the good results for the unseen classes come from the large number of collected images which unambiguously display the unseen objects, as opposed to the S3DIS case.

Finally, on SemanticKITTI, we reach again similar per-



ResNet-18 [30] (full supervis.)

ResNet-50 [12] (self-supervis.)

(a) S3DIS

(b) S3DIS

(c) ScanNet

(d) ScanNet

(e) SemanticKITTI (main split)

(f) SemanticKITTI (main split)

Figure 14. t-SNE [72] visualizations of image-based embeddings for the semantic segmentation datasets (**blue seen**, **orange unseen**).

formance for all types of class representations for most of the seen classes. Among the unseen classes, the mIoU drops significantly for the classes '*Bicyclist*' and '*Motorcycle*' when using the image-based representations whereas it doubles for the class '*Truck*'. A possible explanation for the drop is that many images of the class '*Motorcycle*' actually shows someone riding the motorcycle, which is considered as class '*Motorcyclist*' in SemanticKITTI. A similar phenomenon is observed for classes '*Bicycle*' and '*Bicyclist*'. It also probably explain why the t-SNE representation of these pairs of classes are close to each other (see Figure 14).

D.4. Visualisation of class prototype spaces

Figure 14 shows t-SNE visualizations [72] of the class prototypes extracted for the three datasets.

We observe the same clusters for both kinds of pre-trained networks, which confirms that the difference in pre-training only has a somehow marginal impact on the results. (Please also remember that t-SNE visualization is not deterministic.) Besides, these diagrams remain consistent with the groupings already observed with word embeddings (cf. Figure 12), although they slightly differ.

D.5. Sensitivity to the image collection quality

To evaluate the impact of bad images in the image collections harvested automatically, we manually removed images that were not correct instances of the desired classes. As we are only removing images and not adding new ones, the image collections are smaller after this process. Consequently, it may have both a positive and a negative effect.

Results are shown in Table 16. We notice a significant improvement on SemanticKITTI, where the HmIoU more than doubles. We also notice an improvement on the other datasets, except a slight drop of performance for ScanNet when using ResNet-50, possibly due to reduction of the number of images. This experiment confirms that finding images that unambiguously represents the object category is key in reaching a good performance.

	ResNet-18 [30]		ResNet-50 [12]	
	(full supervis.)		(self-supervis.)	
Dataset	Original	Denoised	Original	Denoised
S3DIS	5.7	**6.5**	4.1	**7.9**
ScanNet	13.9	**15.5**	**15.5**	14.7
SemanticKITTI	3.6	**8.2**	5.3	**11.1**

Table 16. Impact on HmIoU (%) when "denoising" the image collections, using both kinds of pre-trained networks.

References

[1] Ziad Al-Halah, Makarand Tapaswi, and Rainer Stiefelhagen. Recovering the missing link: Predicting class-attribute associations for unsupervised zero-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5975–5984, 2016. 2

[2] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3D semantic parsing of large-scale indoor spaces. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1534–1543, 2016. 1, 4

[3] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019. 1, 4

[4] Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. Generalized denoising auto-encoders as generative models. In *Advances in neural information processing systems*, pages 899–907, 2013. 2, 5, 9

[5] Alexandre Boulch. ConvPoint: Continuous convolutions for point cloud processing. *Computers & Graphics*, 88:24 – 34, 2020. 2, 4, 10

[6] Alexandre Boulch, Joris Guerry, Bertrand Le Saux, and Nicolas Audebert. SnapNet: 3D point cloud semantic labeling with 2D deep segmentation networks. *Computers & Graphics*, 71:189–198, 2018. 2

[7] Alexandre Boulch, Gilles Puy, and Renaud Marlet. FKAConv: Feature-kernel alignment for point cloud convolution. In *Asian Conference on Computer Vision (ACCV)*, 2020. 4

[8] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017. 2

[9] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representation (ICLR)*, 2014. 2

[10] Maxime Bucher, Stephane Herbin, and Frederic Jurie. Generating visual representations for zero-shot classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*, 10 2017. 2, 3, 5, 9, 10, 14

[11] Maxime Bucher, Tuan-Hung Vu, Matthieu Cord, and Patrick Pérez. Zero-shot semantic segmentation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 1, 2, 3, 5, 7, 9, 14

[12] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems*, 33, 2020. 8, 20, 21, 22

[13] Wei-Lun Chao, Soravit Changpinyo, Boqing Gong, and Fei Sha. An empirical study and analysis of generalized zero-shot learning for object recognition in the wild. In *European Conference on Computer Vision*, pages 52–68. Springer, 2016. 4

[14] Jiaxin Cheng, Soumyaroop Nandi, Prem Natarajan, and Wael Abd-Almageed. Sign: Spatial-information incorporated generative network for generalized zero-shot semantic segmentation. *ICCV2021*, 2021. 1, 2

[15] Ali Cheraghian, Shafin Rahman, Dylan Campbell, and Lars Petersson. Mitigating the hubness problem for zero-shot

learning of 3d objects. *arXiv preprint arXiv:1907.06371*, 2019. 1, 2, 4, 6, 10, 11, 12, 13, 16

[16] Ali Cheraghian, Shafin Rahman, Dylan Campbell, and Lars Petersson. Transductive zero-shot learning for 3D point cloud classification. In *IEEE Winter Conference on Applications of Computer Vision*, pages 923–933, 2020. 1, 2, 4, 6, 10, 12, 13

[17] Ali Cheraghian, Shafinn Rahman, Townim F Chowdhury, Dylan Campbell, and Lars Petersson. Zero-shot learning on 3d point cloud objects and beyond. *arXiv preprint arXiv:2104.04980*, 2021. 2, 4, 10, 13

[18] Ali Cheraghian, Shafin Rahman, and Lars Petersson. Zero-shot learning of 3d point cloud objects. In *16th International Conference on Machine Vision Applications (MVA)*, pages 1–6. IEEE, 2019. 1, 2, 4, 5, 6, 7, 8, 10, 12, 13, 14

[19] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 4

[20] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3844–3852, 2016. 2

[21] Ali Farhadi, Ian Endres, Derek Hoiem, and David Forsyth. Describing objects by their attributes. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1778–1785. IEEE, 2009. 2

[22] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. Devise: A deep visual-semantic embedding model. In *Advances in neural information processing systems*, pages 2121–2129, 2013. 2, 6, 7, 8

[23] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012. 4

[24] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning (ICML)*, 2017. 2

[25] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3D semantic segmentation with submanifold sparse convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9224–9232, 2018. 2

[26] Zhangxuan Gu, Siyuan Zhou, Li Niu, Zihan Zhao, and Liqing Zhang. Context-aware feature generation for zero-shot semantic segmentation. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 1921–1929, 2020. 2

[27] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777, 2017. 2

[28] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from RGB-D images for object detection and segmentation. In *European Conference on Computer Vision (ECCV)*, pages 345–360. Springer, 2014. 2

[29] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum Contrast for Unsupervised Visual Representation Learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9726–9735, 2020. 19

[30] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 8, 20, 21, 22

[31] Ping Hu, Stan Sclaroff, and Kate Saenko. Uncertainty-aware learning for zero-shot semantic segmentation. *NeurIPS*, 3:5, 2020. 1, 2

[32] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Pointwise convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 984–993, 2018. 2

[33] Dinesh Jayaraman and Kristen Grauman. Zero-shot recognition with unreliable attributes. In *Advances in neural information processing systems*, pages 3464–3472, 2014. 2

[34] Naoki Kato, Toshihiko Yamasaki, and Kiyoharu Aizawa. Zero-shot semantic segmentation via variational mapping. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019. 2

[35] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *preprint arXiv:1412.6980*, 2014. 9

[36] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Machine Learning (ICML)*, 2017. 2

[37] Vinay Kumar Verma, Gundeep Arora, Ashish Mishra, and Piyush Rai. Generalized zero-shot learning via synthesized examples. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4281–4289, 2018. 2

[38] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 951–958. IEEE, 2009. 2

[39] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4558–4567, 2018. 2

[40] Felix Järemo Lawin, Martin Danelljan, Patrik Tosteberg, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Deep projective 3D semantic segmentation. In *International Conference on Computer Analysis of Images and Patterns (CAIP)*, pages 95–107, 2017. 2

[41] Jiaxin Li, Ben M Chen, and Gim Hee Lee. SO-Net: Self-organizing network for point cloud analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9397–9406, 2018. 2

[42] Peike Li, Yunchao Wei, and Yi Yang. Consistent structural relation learning for zero-shot segmentation. *Advances in Neural Information Processing Systems*, 33, 2020. 1, 2

[43] Yujia Li, Kevin Swersky, and Rich Zemel. Generative moment matching networks. In *International Conference on Machine Learning*, pages 1718–1727, 2015. 2, 5, 9

[44] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. In *International Conference on Learning Representations (ICLR)*, 2016. 2

[45] 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 I. Pratikakis, M. Spagnuolo, T. Theoharis, L. Van Gool, and R. Veltkamp, editors, *Eurographics Workshop on 3D Object Retrieval*. The Eurographics Association, 2015. 12, 13

[46] Bo Liu, Qiulei Dong, and Zhanyi Hu. Segmenting 3d hybrid scenes via zero-shot learning. *arXiv preprint arXiv:2107.00430*, 2021. 2, 14

[47] Jingen Liu, Benjamin Kuipers, and Silvio Savarese. Recognizing human actions by attributes. In *CVPR 2011*, pages 3337–3344. IEEE, 2011. 2

[48] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8895–8904, 2019. 2

[49] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015. 2

[50] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015. 2, 5, 9

[51] Daniel Maturana and Sebastian Scherer. VoxNet: A 3D convolutional neural network for real-time object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928, 2015. 2

[52] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013. 2, 5

[53] Mohammad Norouzi, Tomas Mikolov, Samy Bengio, Yoram Singer, Jonathon Shlens, Andrea Frome, Greg S Corrado, and Jeffrey Dean. Zero-shot learning by convex combination of semantic embeddings. *arXiv preprint arXiv:1312.5650*, 2013. 2, 7

[54] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *International conference on machine learning*, pages 2642–2651, 2017. 2, 5, 9

[55] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. 2, 5

[56] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 3, 6

[57] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5648–5656, 2016. 2

[58] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5105–5114, 2017. 2

[59] Milos Radovanovic, Alexandros Nanopoulos, and Mirjana Ivanovic. Hubs in space: Popular nearest neighbors in high-dimensional data. *Journal of Machine Learning Research*, 11(sept):2487–2531, 2010. 2

[60] Mahdi Rezaei and Mahsa Shahidi. Zero-shot learning and its applications from autonomous vehicles to covid-19 diagnosis: A review. *arXiv preprint arXiv:2004.14143*, 2020. 1, 2

[61] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. OctNet: Learning deep 3D representations at high resolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3577–3586, 2017. 2

[62] Xavier Roynard, Jean-Emmanuel Deschaud, and François Goulette. Classification of point cloud scenes with multiscale voxel deep network. *arXiv preprint arXiv:1804.03583*, 2018. 2

[63] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 8, 18

[64] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008. 2

[65] Edgar Schonfeld, Sayna Ebrahimi, Samarth Sinha, Trevor Darrell, and Zeynep Akata. Generalized zero-and few-shot learning via aligned variational autoencoders. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8247–8255, 2019. 2, 6, 13

[66] Yutaro Shigeto, Ikumi Suzuki, Kazuo Hara, Masashi Shimbo, and Yuji Matsumoto. Ridge regression, hubness, and zero-shot learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 135–151. Springer, 2015. 2

[67] Kaleem Siddiqi, Juan Zhang, Diego Macrini, Ali Shokoufandeh, Sylvain Bouix, and Sven Dickinson. Retrieving articulated 3-d models using medial surfaces. *Machine vision and applications*, 19(4):261–275, 2008. 12, 13

[68] Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. Zero-shot learning through cross-modal transfer. In *Advances in neural information processing systems*, pages 935–943, 2013. 2

[69] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks

for 3D shape recognition. In *IEEE International Conference on Computer Vision (ICCV)*, pages 945–953, 2015. 2

[70] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3D. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3887–3896, 2018. 2

[71] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. KPConv: Flexible and deformable convolution for point clouds. In *IEEE International Conference on Computer Vision (ICCV)*, 2019. 2, 5, 10

[72] Laurens Van Der Maaten. Accelerating t-sne using tree-based algorithms. *The Journal of Machine Learning Research*, 15(1):3221–3245, 2014. 18, 21, 22

[73] Qian Wang and Ke Chen. Alternative semantic representations for zero-shot human action recognition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 87–102. Springer, 2017. 8

[74] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2589–2597, 2018. 2

[75] Wei Wang, Vincent W Zheng, Han Yu, and Chunyan Miao. A survey of zero-shot learning: Settings, methods, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–37, 2019. 1, 2

[76] Wenxuan Wu, Zhongang Qi, and Li Fuxin. PointConv: Deep convolutional networks on 3D point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9621–9630, 2019. 2

[77] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015. 2, 4, 5, 12, 13

[78] Yongqin Xian, Zeynep Akata, Gaurav Sharma, Quynh Nguyen, Matthias Hein, and Bernt Schiele. Latent embeddings for zero-shot classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 69–77, 2016. 2

[79] Yongqin Xian, Subhabrata Choudhury, Yang He, Bernt Schiele, and Zeynep Akata. Semantic projection network for zero- and few-label semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[80] Yongqin Xian, Subhabrata Choudhury, Yang He, Bernt Schiele, and Zeynep Akata. Semantic projection network for zero-and few-label semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8256–8265, 2019. 1, 2

[81] Yongqin Xian, Christoph H Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE transactions on pattern analysis and machine intelligence*, 41(9):2251–2265, 2018. 1, 2

[82] Yongqin Xian, Tobias Lorenz, Bernt Schiele, and Zeynep Akata. Feature generating networks for zero-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5542–5551, 2018. 2, 6, 13

[83] Yongqin Xian, Saurabh Sharma, Bernt Schiele, and Zeynep Akata. f-vaegan-d2: A feature generating framework for any-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10275–10284, 2019. 2

[84] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. SpiderCNN: Deep learning on point sets with parameterized convolutional filters. In *European Conference on Computer Vision (ECCV)*, pages 87–102, 2018. 2

[85] Li Zhang, Tao Xiang, and Shaogang Gong. Learning a deep embedding model for zero-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2021–2030, 2017. 2

[86] Hang Zhao, Xavier Puig, Bolei Zhou, Sanja Fidler, and Antonio Torralba. Open vocabulary scene parsing. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2002–2010, 2017. 2