GraVoS: Voxel Selection for 3D Point-Cloud Detection

Oren Shrout
Technion, Israel
shrout.oren@gmail.com

Yizhak Ben-Shabat
Technion, Israel & ANU, Australia
sitzikbs@technion.ac.il

Ayellet Tal
Technion, Israel
ayellet@ee.technion.ac.il

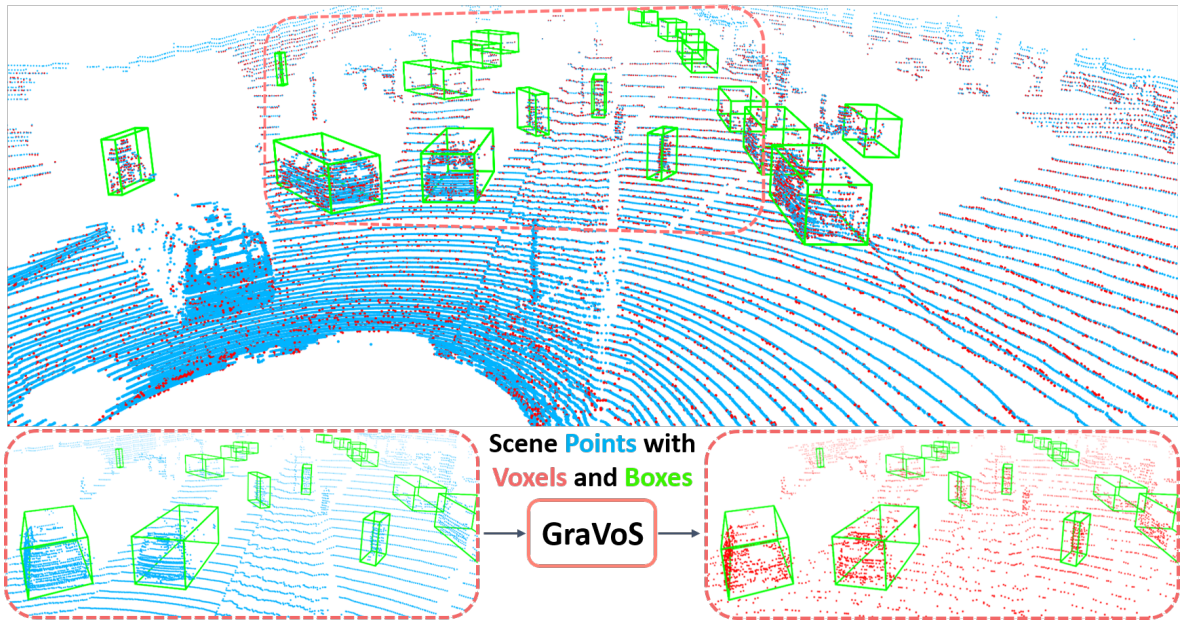


Figure 1. **GraVoS for 3D object detection**. Given a 3D point cloud (cyan) and its associated voxels, we propose a method that selects a subset of network-dependent meaningful voxels (salmon). Our method selects most of the voxels of the challenging classes, those with relatively few training instances (such as the *Cyclists* and *Pedestrians*), less from the prevalent classes (e.g. *Cars*) and very few from the background. It is shown that considering only this subset improves the performance of numerous SoTA voxel-based detectors.

Abstract

3D object detection within large 3D scenes is challenging not only due to the sparsity and irregularity of 3D point clouds, but also due to both the extreme foreground-background scene imbalance and class imbalance. A common approach is to add ground-truth objects from other scenes. Differently, we propose to modify the scenes by removing elements (voxels), rather than adding ones. Our approach selects the "meaningful" voxels, in a manner that addresses both types of dataset imbalance. The approach is general and can be applied to any voxel-based detector, yet the meaningfulness of a voxel is network-dependent. Our voxel selection is shown to improve the performance of several prominent 3D detection methods.

1. Introduction

3D object detection has gained an increasing importance in both industry and academia due to its wide range of applications, including autonomous driving and robotics [8, 33]. LiDAR sensors are the de-facto standard acquisition devices for capturing 3D outdoor scenes (Fig. 1). They produce sparse and irregular 3D point clouds, which are used in a variety of scene perception and understanding tasks.

There are three prominent challenges in outdoor point cloud datasets for detection. The first is the small size of the datasets, in terms of the number of scenes. The second is the large number of points in the scene vs. the small number of points on the training examples (objects). A single scene might contain hundreds of thousands, or even millions, of points but only a handful of objects. The third is class imbalance, where some classes might contain significantly

more instances than the other classes. This often results in lower predictive accuracy for the infrequent classes [12, 18].

To handle the first challenge, it was proposed to enrich the dataset during training with global operations, applied to the whole point cloud, such as rotation along the Z-axis, random flips along the X-axis, and coordinate scaling [35, 37, 45]. We note that most methods that solve the second and the third challenges indirectly also solve this first challenge.

To solve the second challenge, it is suggested to augment the scene by local operations, applied to points belonging to individual objects [4, 5]. Local operations include random point drop out, frustum drop out, additive noise added to the object, intra-object mixing, and swapping regions between different objects.

To solve the third challenge, as well as the second one, [24–26, 35] propose to add ground-truth objects from different point-clouds to the scene, for training. This type of augmentation indeed mitigates the imbalance. However, it does not take into account the network architecture, though Reuse *et al*. [23] show, through a series of experiments, that both local and global data augmentation for 3D object detection strongly depends not only on the dataset, but also on the network architecture. Thus, network-dependent augmentations are beneficial.

We present a novel network-dependent data modification approach that addresses the latter two challenges. (We address the first challenge indirectly, similarly to [24–26].) The key idea is to learn a subset of elements of the scene, which are meaningful for object detection. Inline with [23]'s observation, meaningfulness is defined in the context of the network and not only of the scene. Considering only this subset as input will allow us not only to decrease the number of elements in the scene, but also to increase the class balance within a given scene.

To realize this idea, we focus on SoTA detection networks that transform point clouds into voxels as a first step in their pipeline. The main reason behind transferring the input to voxels is reducing the size of the input, as only the occupied voxels are then processed. This enables these systems to work on extremely large scenes. Obviously, another benefit of voxels is the ability to impose structure on the input.

Generally speaking, in this voxel-based setup, meaningful voxels are those for which the model "struggles" to locate the objects. Hence, the gradients play a major role in determining the meaningful voxels. Our approach is thus termed *Gradient-based Voxel Selection (GraVoS)*.

We show that when focusing only on the meaningful voxels and removing the non-informative ones, most of the discarded voxels belong to the scene background. Few of the removed voxels are associated with the prevalent classes and almost none are associated with the non-prevalent classes. Our strategy may be contrasted with *dropout* augmentation techniques, which reduce the number of elements randomly [4, 5].

Our distribution balancing is demonstrated in Fig. 1. Given a point cloud (cyan), our selected subset of the meaningful voxels (salmon) contains significantly few points from the background (the points are voxel centers here), more points on the objects that belong to the *Car* class (the most prevalent class), and almost all the points on the objects that belong the *Pedestrian* and the *Cyclist* classes.

Our method is general and may be applied to different voxel-based networks. Furthermore, it comes at no additional inference time cost. We show results on four SoTA networks: SECOND [35], Part-A^2 [26], Voxel R-CNN [6], and CenterPoint [42] on the well-established KITTI dataset [7, 8]. The performance of all networks improves, in particular when considering the difficult categories of *Pedestrian* and *Cyclist*. For instance, the performance of [26] on the benchmark's moderate subset improves by 2.32% & 1.15% for the non-prevalent classes (*Cyclist* and *Pedestrian*), which constitutes an error reduction of 8.20% & 2.77%, respectively.

Hence, the main contributions of this paper are:

- A novel & generic "meaningful" voxel selection method, called *Gradient-based Voxel Selection*.
- A training procedure that uses the selected voxels to improve 3D detection without additional data. This procedure combines information from different stages of the model's training.
- Demonstrating improved performance of four voxel-based SoTA detection methods, successfully coping both with the inherent class imbalance and with the foreground-background imbalance.

2. Related Work

3D Object Detection. Object detection methods aim at localizing objects in a given scene and classifying them. 3D detection methods can be categorized into grid-based [3, 6, 14, 26, 35–37, 41–45] and point-based methods [24, 25, 27, 38, 40]. See [11] for an excellent survey on deep learning for point clouds in general and for detection in particular.

Grid-based methods first transform the given point cloud into a regular representation, either voxels [3, 6, 14, 26, 35, 41–45] or a 2D Bird-Eye View (BEV) [36, 37]. This enables processing using 3D or 2D CNNs, respectively. The resulting voxels, however, have a very sparse spatial distribution [35]. To handle sparsity, sparse convolutions [9, 10] have been proposed. Modifications of the sparse convolution were proposed by [35] for efficient feature extraction and by [6, 24, 26] for efficiently generating box proposals. These grid-based approaches provide efficient and accurate solutions, but are limited when the data is imbalanced.

For point-based approaches, point set abstractions are learned directly from the raw point cloud, by utilizing

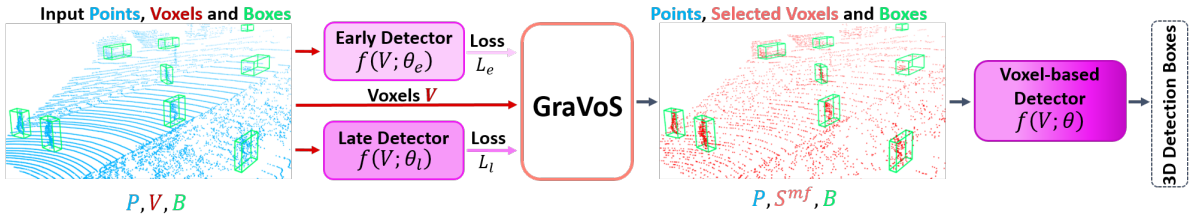


Figure 2. **Training with GraVoS**. An input point cloud (cyan) is voxelized and fed into a pre-trained voxel-based detector at two different training stages, early and late (with frozen weights). These detectors' losses are computed and are the input of GraVoS, which performs voxel selection. The selected voxels (salmon) are then fed into the late detector, initializing its weights where it left off (θ_l) and continuing training using the selected voxels exclusively. Here, $f(\cdot)$ is a voxel-based network.

PointNet-like architectures [20, 21]. For instance, PointRCNN [25] uses the PointNet backbone to generate proposals directly from the point cloud for a refinement stage. STD [40] further improves the refinement stage by densifying the sparse proposal. A fusion sampling strategy of [38] takes into account the distances between the points in both coordinate and feature spaces. While point-based methods are quantization-free and have flexible receptive fields, they suffer from GPU memory limitations and from a high computation time when processing large point clouds.

Augmentation methods for 3D detection. Since the size of the available training data is limited, most SoTA 3D detection methods use a data augmentation protocol to alleviate the overfitting problem [3, 6, 24–26, 35, 38, 41, 45]. Generally speaking, global operations are applied to the whole point cloud scene, including random coordinate scaling, random flips along the X-axis, and random rotations around the Z-axis. Despite their benefits, these augmentations do not address the challenges of data imbalance. In [35] it is proposed to randomly insert cropped ground-truth objects from different scenes of the training data, while avoiding overlapping objects.

Recent works suggest to augment the training scenes also by local operations on individual objects. These operations can be subdivided into two categories: subtractive and additive. For the subtractive operations, it is suggested in [4] to apply random points or frustum drop out. For the additive operations, it is proposed in [4] to add noise to objects' points and in [5] to also apply mixing or swapping parts between different objects in the same class.

Alternative methods including bootstrapping, or *hard negative mining*, have been used to handle data imbalance [17, 22, 28, 31]. The key idea is to gradually add examples to the training set, for which the detector classifier gets false positives. This is usually done iteratively, where the detector is first trained on the whole training set and then the training set is updated based on the detector's new false positives. Differently from these approaches, our training set is not explicitly updated using the ground truth information.

3. Gradient-based Voxel Selection (GraVoS)

We proposes a novel subtractive approach, which provides a further boost in performance to voxel-based 3D detection methods. We focus on two properties that make typical scenes challenging for detection systems: (1) foreground-background imbalance and (2) class imbalance.

Given an input point-cloud of an outdoor scene, 3D detectors aim to localize objects and classify them. We concentrate on voxel-based detectors, since they are beneficial in handling scenes with a very large number of points. Typically, these detectors start by converting the cloud into a voxel grid. Then, non-occupied voxels are removed and the points in every occupied voxel are grouped. To address the varying number of points in the voxels, random sampling of points is applied to each voxel. These voxels are then fed into a detection network, which is usually a sparse 3D CNN, followed by a region proposal network (RPN). The detector outputs both bounding box proposals and class predictions. While this approach exhibits good performance for the prevalent classes, it might deteriorate for other classes.

Training with GraVoS. We propose to add a selection component to the above general approach. The goal is to select the "meaningful" voxels and to remove the less meaningful ones from the scene, in a manner that addresses the two imbalance challenges discussed above. Meaningfulness is indicated by the magnitude of the gradients, as the gradients encapsulate information regarding the detector's success. Selecting the voxels with high magnitudes puts more emphasis on the voxels that determine the locations of the objects.

Fig. 2 illustrates the training of an existing 3D voxel-based detector with our *Gradient-based Voxel Selection (GraVoS)* module. First, a voxel detector $f(\cdot)$ is trained on the dataset without any modification. Its parameters are stored at two different training stages – early stage and late stage, θ_e and θ_l, respectively. Then, the voxels are fed into these pre-trained detectors, $f(V;\theta_e)$, $f(V;\theta_l)$, separately. Each detector's location loss is fed into our GraVoS module, along with the computational graph and the voxelized scene. Within the GraVoS module, the meaningful voxels of the

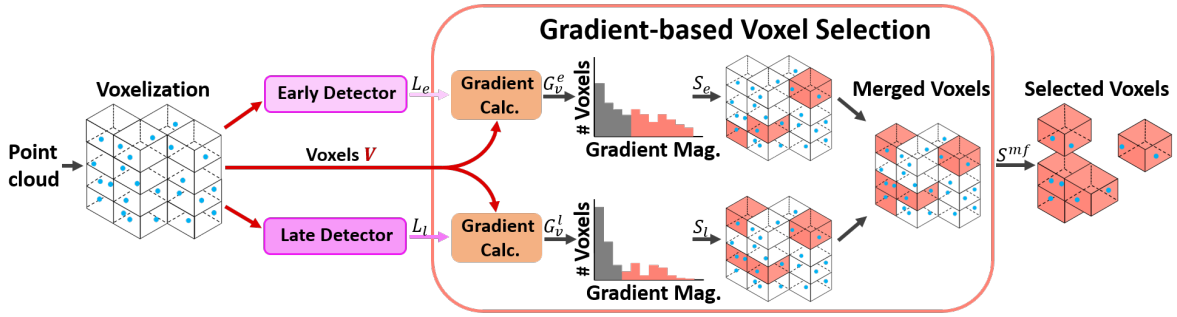


Figure 3. **GraVoS Module.** The voxelized point cloud is fed into the GraVoS module and the pre-trained detector (at two training stages). The detectors' losses are computed and fed into the GraVoS module. These losses are used to compute the gradient magnitude at each voxel. For each detector stage the voxels are selected based on their gradients' magnitude. The selected voxels (highlighted in salmon) from the two stages are then merged to form the final selected subset of voxels, S^{mf}.

voxelized scene are found. These become the input to a new copy of the late stage detector $f(V;\theta)$, which is further trained using *only* this refined subset.

We use two different training stages since they provide complementary information regarding voxel meaningfulness. The late stage detector assigns higher values to meaningful and unique voxels and lower values to meaningful voxels in repetitive and easy to learn features. Conversely, in the early stage detector, these "easy" features still maintain a high gradient magnitude. Though learned early on, they are essential for recognizing the objects. Hence, merging the two sets is beneficial, as shown in the ablation study.

GraVoS Module. GraVoS aims at selecting the meaningful voxels and discarding the less informative ones. Its structure is illustrated in Fig. 3. A voxelized grid of the point cloud scene (cyan) and the detector location losses from the early and the late stages, as well as the computational graphs, are fed into GraVoS. Then, for each detector's loss, the gradients' magnitude are calculated and the meaningful voxels are found (salmon). Finally, the voxels that pass the threshold for each detector's stage are merged, to create the selected voxel set. We elaborate on the selection process hereafter.

The magnitudes of the gradients of the losses w.r.t. each input point (each voxel contains its points) is computed. These are then aggregated for all the points in the voxel, to get a scalar value per voxel. This value will be later used as an informativeness measure. Formally, let L_e, L_l denote the computed location losses for the early and the late stages, respectively. (These losses are associated with the specific detector and are defined accordingly; see the supplementary for the exact definitions.) The gradient magnitude per voxel, v_i, is computed as:

$$G^e_{v_i} = \frac{1}{n_i} \sum_{p^i_j \in v_i} \left\| \frac{\partial L_e}{\partial p^i_j} \right\|, \; G^l_{v_i} = \frac{1}{n_i} \sum_{p^i_j \in v_i} \left\| \frac{\partial L_l}{\partial p^i_j} \right\|, \quad (1)$$

where n_i is the number of points, p^i_j, in voxel v_i.

For each detector stage, early and late, we use the gradient magnitude to create a subset of meaningful voxels S_e and S_l, respectively. For the late stage detector we assign the voxels with the *top-k* magnitude values $G^l_{v_i}$ to S_l. For the early stage detector we assign voxels with magnitude values $G^e_{v_i}$ larger than the mean $\overline{G^e_{v_i}}$ to S_e. The different threshold mechanisms are due to the fact that at the early stages the high gradients are noisy, thus we should not consider only the largest gradients. We show the benefit of the different mechanisms in the ablation study. Formally, the subsets are defined in Equations (2) to (3):

$$S_e = \{v_i \mid G^e_{v_i} \geq \overline{G^e_{v_i}}\}, \quad (2)$$

$$S_l = \{v_i \mid G^l_{v_i} \in \textit{top-k}(G^l_{v_i})\}. \quad (3)$$

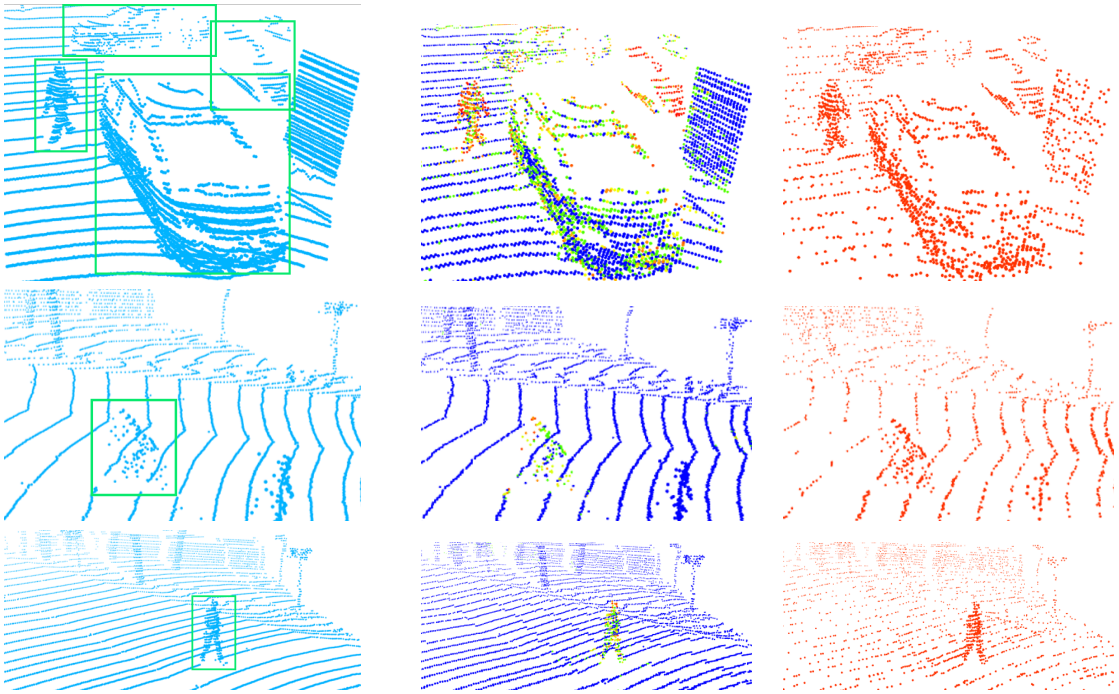
The parameter k gives control over the percentage of voxels that are considered meaningful. It is selected based on two variables, (1) n_{vs}, the number of meaningful voxels in the final selected set (S^{mf}) and (2) ν_{idr}, the intra-detector ratio between the late and the early detectors. It is calculated as $k = \nu_{idr} \cdot n_{vs}$. Both n_{vs} and ν_{idr} are hyper parameters. (In practice, they are set to 80% of the number of input voxels and to $50/80$, respectively).

Next, we merge the subsets above by a union operator, to form the final meaningful voxels subset:

$$S^{mf} = S_l \cup S_e. \quad (4)$$

Finally, the selected voxels of S^{mf} are fed into the pre-trained detector $f(V;\theta)$, which is fine tuned for several epochs using the detector's original losses. Note that GraVoS does not affect the inference time, since it is only applied at training.

Fig. 4 depicts the gradient magnitudes and the resulting S^{mf} for a pre-trained detector (Voxel R-CNN [6]) for the three object classes of KITTI's benchmarks: *Car, Cyclist*, and *Pedestrian*. The background voxels have low gradient magnitudes (b), making them less likely than foreground



(a) Input (GT objects are in rectangles) (b) Gradinent magnitudes (c) Final subset S^{mf}

Figure 4. **Gradient-based voxel selection.** Given an input point cloud (a), the magnitudes of the gradients are computed (b), and the selected voxel subset is computed (c). The magnitude of the gradients is depicted as a colormap from blue to red (low to high values). Evidently, the gradients on the background voxels are lower and are therefore less likely to be selected than foreground pixels. The objects' voxels have high gradients and therefore most of their voxels are retained in the final subset. However, there are differences between the classes: The less prominent classes, *Cyclist* (middle) and *Pedestrian* (bottom) retain relatively more points than the prominent class *Car* (top).

voxels to be selected for training the detector (c). Conversely, many more points on the objects are maintained thanks to their high gradient magnitude. Almost all the voxels of the *Cyclist* and *Pedestrian* objects have high gradients, and are therefore selected to the final subset, compared to the *Car* objects, with a somewhat smaller subset.

GraVoS for two-stage detectors. Generally speaking, 3D object detectors can be grouped by the number of stages in their detection pipeline, into single-stage [14, 35–38, 42, 45] or two-stage detectors [2, 3, 6, 13, 15, 19, 24–26, 34, 39, 40]. Single-stage approaches are fast since they usually have a single feed-forward network to predict the bounding boxes and classes. However, their main drawback is accuracy, since there is no component that specializes and fine tunes the box orientations. The two-stage approaches have an additional refinement module. This makes them slower and heavier in memory, but their accuracy is improved.

GraVoS is general in the sense that it can be used for both single-stage and two-stage voxel-based detectors. However, the refinement stage (second stage) requires local information from the original input (point/voxels), which is not available after the selection process. As illustrated in Fig. 5, in this case, we apply GraVoS only on the first stage and transfer the missing local information (from the late stage detector) to the refinement stage. In our experiments (Section 4), we show the benefits of GraVoS for both single stage [35, 42] and two stage [6, 26] approaches.

4. Experiments

To evaluate the performance of our method, in Section 4.1 we compare SoTA 3D object detection methods on the well established KITTI dataset [8], with and without our GraVoS module and training procedure. In addition, in Section 4.2 we explore several design choices for GraVoS.

KITTI dataset. KITTI [8] is the most widely-used 3D object detection dataset for autonomous driving. Its training set contains 7481 examples that are divided into a training subset, with 3712 examples, and a validation subset, with 3769 examples [1]. The test set contains 7518 examples. We report results for all three classes in KITTI's benchmarks, *Car*, *Pedestrian* and *Cyclist*, which contain 28742, 4487, and 1627 object instances, respectively.

Evaluation metrics. We report our results using the corrected average precision (AP) metric of Equation 5, which

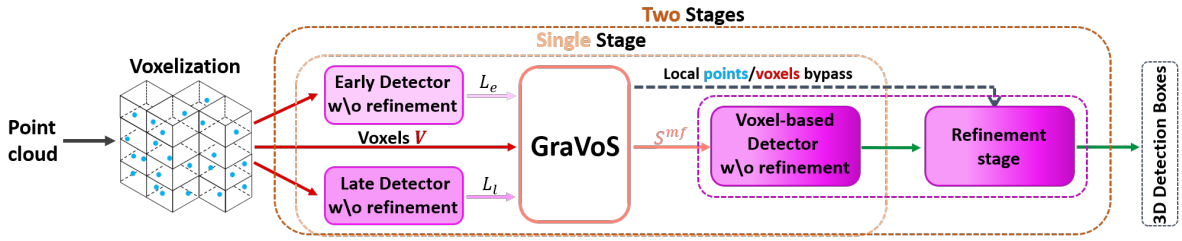


Figure 5. **Incorporating GraVoS into two-stage detectors**. Voxel selection is performed during the first stage, as before. Since in two-stage architectures, the detector consists of a proposal generator and a refinement module, we use the detector without the refinement component in the first stage. The last refinement stage (in the second stage) gets the output of the proposal generator (the green arrow), as well as the required local data that is bypassed through GraVoS.

Method	*Pedestrian*			*Cyclist*			*Car*			Average			
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	*Ped.*	*Cyc.*	*Car*	All
Voxel R-CNN [6]	66.94	59.88	54.95	89.83	72.49	**68.87**	**92.62**	85.13	82.73	60.59	77.06	86.83	74.83
[6]+Ours	**68.52**	**61.63**	**56.71**	**91.97**	**72.98**	68.37	92.40	**85.41**	**82.84**	**62.29**	**77.77**	**86.88**	**75.65**
SECOND [35]	54.90	49.84	45.15	81.85	65.42	61.26	**90.79**	**81.87**	**78.75**	49.96	69.51	**83.80**	67.76
[35]+Ours	**57.75**	**51.99**	**47.54**	**84.36**	**66.41**	**62.61**	89.53	81.06	78.07	**52.43**	**71.13**	82.89	**68.81**
Part-A^2 [26]	65.37	58.43	53.62	89.45	71.71	67.74	**91.88**	**82.64**	80.21	59.14	76.30	84.91	73.45
[26]+Ours	**65.82**	**59.58**	**54.55**	**90.64**	**74.03**	**69.64**	91.68	82.58	**81.67**	**59.98**	**78.10**	**85.31**	**74.47**
CenterPoint [42]	56.85	53.17	49.73	80.27	62.85	60.13	**89.58**	**82.09**	**79.58**	53.25	67.75	**83.75**	68.25
[42]+Ours	**58.02**	**54.64**	**50.94**	**83.40**	**64.81**	**61.42**	88.74	81.74	79.53	**53.53**	**69.88**	83.34	**69.25**

Table 1. **Performance on the 3D detection benchmark.** Each method's performance is compared with and without our voxel selection. Results are reported for the Easy, Moderate (Mod.) and Hard categories on the three classes. Evidently, GraVoS always improves the average performance. Furthermore, it improves the performance of the methods for the non-prevalent classes, while it might slightly degrade the performance for the prevalent class.

is the standard for evaluating 3D detection [29]:

$$AP|_R = \frac{1}{|R|} \sum_{r \in R} \max_{r':r' \geq r} \rho(r'). \tag{5}$$

Here, $\rho(r)$ is the precision at recall r, $R = [1/40,\ 2/40,\ \ldots,\ 1]$ and $|R| = 40$. For precision and recall we use the standard IoU thresholds of $0.7, 0.5, 0.5$ for the *Car*, *Pedestrian*, and *Cyclist* classes, respectively.

The evaluation on KITTI is divided into three difficulty categories: *Easy*, *Moderate* and *Hard*, based on the occlusion, the truncation and the object's distance from the scanner. The more distant and more occluded an object is, the harder it is to be detected.

4.1. Results

To demonstrate the generality and the effectiveness of our method, we show that continuing to train four prominent 3D voxel-based object detectors with GraVoS selection yields improved performance on the challenging classes. The four detectors are SECOND [35], Voxel R-CNN [6], Part-A^2 [26] and CenterPoint [42]. For each of these methods, we use its own protocol of augmentation.

Table 1 reports the results for the 3D detection benchmark and Table 2 reports the results for the Bird-Eye View (BEV) detection benchmark. GraVoS improves the overall performance of all the detectors. Furthermore, GraVoS is especially beneficial for the non-prevalent classes i.e., *Cyclist* and *Pedestrian*, but might slightly degrade the results of the prevalent class *Car*. This is attributed to the selection process, which prefers the non-prevalent classes.

4.2. Ablation Study

This section studies alternatives to GraVoS choices. In the following experiments, SECOND [35] is used as the baseline 3D object detector.

Voxel-selection alternatives. Fig. 6 shows that GraVoS significantly outperforms three alternatives: Dropout, BgSampling and InvFreqSampling. *Dropout* randomly selects a subset of the input voxels [4, 5, 30]. It was shown to avoid over-fitting and to improve the performance for many tasks. However, in our particular case, where the number of object voxels is significantly smaller than the total number of scene voxels, the benefit of Dropout is limited. This is so since too few voxels that belong to object classes remain when sampling randomly. See supplemental for additional results.

Interestingly, we further show that our performance is even better than sampling when the ground-truth voxel classes are known (which contradicts our assumption). We

Method	*Pedestrian* Easy	*Pedestrian* Mod.	*Pedestrian* Hard	*Cyclist* Easy	*Cyclist* Mod.	*Cyclist* Hard	*Car* Easy	*Car* Mod.	*Car* Hard	Average *Ped.*	Average *Cyc.*	Average *Car*	Average All
Voxel R-CNN [6]	69.97	63.60	59.04	93.63	76.09	**72.58**	**95.96**	91.43	**90.70**	64.20	80.77	**92.70**	79.22
[6]+Ours	**72.37**	**66.24**	**60.31**	**94.47**	**76.32**	71.60	**95.96**	**91.96**	89.49	**66.31**	**80.80**	92.47	**79.86**
SECOND [35]	60.94	55.73	51.56	87.87	**70.91**	66.57	92.30	**89.68**	**87.51**	56.08	75.12	89.83	73.67
[35]+Ours	**62.23**	**56.78**	**52.63**	**89.02**	70.88	**66.77**	**92.86**	89.62	87.26	**57.21**	**75.56**	**89.91**	**74.23**
Part-A^2 [26]	68.31	61.70	57.33	91.19	75.42	70.97	**92.89**	**90.14**	**88.17**	62.45	79.19	**90.40**	77.35
[26]+Ours	**68.51**	**62.40**	**58.04**	**93.13**	**75.91**	**72.68**	92.85	90.07	88.13	**62.98**	**80.57**	90.35	**77.97**
CenterPoint [42]	61.26	58.08	54.83	83.84	66.40	63.05	**92.26**	**89.30**	**88.10**	58.06	71.10	**89.89**	73.01
[42]+Ours	**62.32**	**59.19**	**55.80**	**85.68**	**68.22**	**64.51**	91.91	88.90	88.00	**59.10**	**72.80**	89.60	**73.84**

Table 2. **Performance on the Bird-Eye View (BEV) detection benchmark.** As in Table 1, our method is beneficial for all four methods.

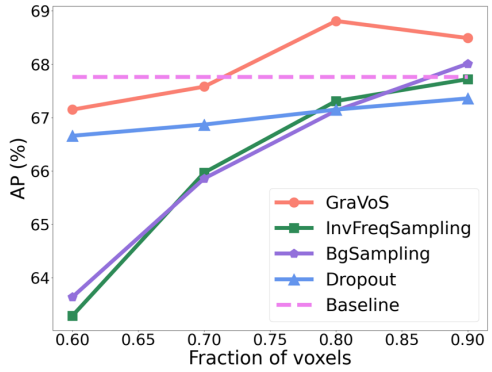
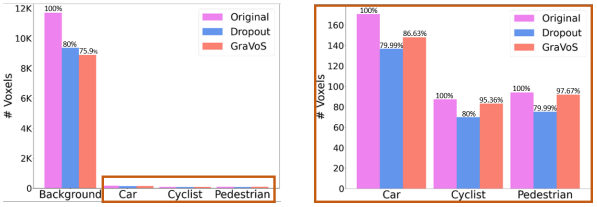


Figure 6. **Comparison to alternative approaches.** GraVoS is compared to Dropout, BgSampling and InvFreqSampling for different voxel selection ratios. The baseline is the constant performance of the detector (all voxels). When too few voxels are used (< 0.7), the detector misses objects, as expected. For ratios larger than 0.7, GraVoS outperforms other approaches significantly. This is due to the fact that we use the meaningful voxels.



(a) background & foreground (b) foreground zoom in

Figure 7. **GraVoS as a data balancer.** This figure shows the average number of voxels with and w/o GraVoS. (a) The background has significantly more voxels than the foreground and accordingly, significantly higher voxel reduction. (b) Zooming into the foreground, the *Car* category receives most of the voxel reduction. Thus, GraVoS effectiveness may be attributed to data balancing.

utilized two classical class sampling techniques, in order to sample more voxels from the foreground (and from less prevalent classes) and less from the background. In *BgSampling* voxels are randomly removed from the background until a threshold is met; in *InvFreqSampling*, sampling is based on the inverse of the class frequencies in a given scene.

Class-level voxel balancing with GraVoS. We analyze GraVoS's effectiveness by inspecting its influence on the average number of voxels for each object category. Fig. 7 shows that while GraVoS reduces the number of voxels in each object class, not all classes exhibit the same reduction. The background voxels are reduced by 24.1%, the *Car* category by 13.37%, while *Cyclist* and *Pedestrian* are hardly affected with only a 4.64% and 2.33% reduction, respectively. Essentially, GraVoS discards relatively more voxels from the background points than from objects, addressing the foreground-background imbalance challenge. This inherently differs from the naive Dropout, where the reduction is uniform across the whole scene (20%). This indicates that GraVoS has an object-level data balancing effect.

Voxel selection ratio. What shall be the number of selected voxels during training, n_{vs}? Since this parameter inherently depends on the number of input voxels, n_v, we explore the selection ratio $\nu_{vs} = \frac{n_{vs}}{n_v}$. Fig. 6 shows that, as expected, it is beneficial to take a high ratio of voxels and that the best ratio is 0.8. This can be explained by the fact that uninformative voxels are discarded from the training process, allowing the network to focus on more informative voxels. However, smaller ratios will cause a large gap between the distributions of the train and the test sets, which would result in an under performance of the detector.

Intra-detector ratio. We study the number of voxels that should be taken from each stage (early and late) of the pre-trained detector. For that, we use the intra-detector ratio ν_{idr}, which quantifies the fraction of voxels from the late stage detector w.r.t. the total number of selected voxels n_{vs}. Recall that when all the voxels are taken from the early-stage detector then $\nu_{idr} = 0$, and when all the voxels are taken from the late-stage detector then $\nu_{idr} = 1$. We start by setting $\nu_{idr} = 30/80$ and get class average accuracy of 67.58%. Then, we increase the intra-detector ratio ν_{idr} by increments of $10/80$, taking more voxels from the late-stage than from the early-stage, up to $\nu_{idr} = 1$, which yields accuracy of 68.48%. The best result is achieved for $\nu_{idr} = 50/80$, with

Epoch	1	5	10	20	30	40
mAP_{3D}	**68.81**	68.59	68.59	68.58	68.34	68.27

Table 3. **Training duration for the early-stage detector.** mAP_{3D} is the average precision over all the classes and difficulty levels for different early-stage detectors. The best result is achieved after a single epoch. This may be attributed to the fact that the gradient magnitudes of easy-to-learn features are still non-negligible.

detection accuracy of 68.81%. (For this experiment we use a fixed voxel selection ratio $\nu_{vs} = 0.8$.)

This study shows that selecting voxels mostly from the early-stage does not suffice for fine tuning the original detector (late-stage). Moreover, it shows that the early-stage indeed provides additional information that the late-stage lacks, when a proper ratio is used.

Early-stage detector's training duration. In our framework, we consider the fully trained detector as the late-stage detector. Therefore, we only have to choose for how long the detector needs to be trained in the early-stage. To this end, we tested different epoch choices for the early-stage. Table 3 shows that the performance is in favor of earlier epochs, where the first epoch achieves the best result. This is expected since after the first epoch, the magnitude of the gradients at meaningful voxels with features that are easy to learn had not yet vanished. However, even for higher epochs using the early-stage is beneficial and achieves better results on average than the original detector (67.76%). (For this ablation study we used $\nu_{vs} = 0.8$ and $\nu_{idr} = 50/80$.)

Early and late detector mechanism choices. We tested three different choices for the early and the late detectors: *mean*, *median*, and *top-k*. For the mean and the median strategies, the voxels selected are those with gradients' magnitude higher than the mean or the median. For the *top-k* strategy, we consider two choices for the intra-detector ratio, $\nu_{idr} = 50/80$ and $\nu_{idr} = 30/80$, where 80% of the total voxels were sampled in this experiment.

Table 4 shows that most of the different choices improve the baseline detector [35]. Selecting the *top-k* with $\nu_{idr} = 50/80$ for the late detector is the most beneficial, especially with the mean mechanism for the early detector.

Implementation details. We use the 3D detector implementations available in the OpenPCDet toolbox [32]. For a fair comparison, we use the default configurations for all detectors. We set the voxel dimensions to be $(0.05, 0.05, 0.1)$, as provided in the toolbox. For fine-tuning with GraVoS, we continue to train for 60 epochs, 40 epochs using the original detector's optimizer and 20 epochs using stochastic gradient decent (SGD) and a step decay optimizer. Note that we use the same settings for all methods [6, 26, 35, 42] after tuning for [35] in our ablation study. In the supplemental, we provide additional results that show that the performance boost is not attributed to longer training or to further scheduler hyper-parameter details. All experiments were done on a single NVIDIA $A100$ GPU.

Early mechanism	Late mechanism	mAP_{3D}
Baseline [35]		67.76
mean	*mean*	68.25
mean	*median*	68.01
mean	*top-k* ($\nu_{idr} = 50/80$)	**68.81**
median	*mean*	67.80
median	*median*	68.37
median	*top-k* ($\nu_{idr} = 50/80$)	68.25
top-k ($\nu_{idr} = 50/80$)	*mean*	67.33
top-k ($\nu_{idr} = 50/80$)	*median*	68.07
top-k ($\nu_{idr} = 50/80$)	*top-k* ($\nu_{idr} = 30/80$)	67.99
top-k ($\nu_{idr} = 30/80$)	*top-k* ($\nu_{idr} = 50/80$)	68.28

Table 4. **Different mechanism choices for the detectors.** Most selection choices for the early and late detectors improve the baseline detector. The best selection combination is the *top-k* with $\nu_{idr} = 50/80$ for the late detector and the mean selection for the early detector. In this experiment 80% of the total voxels were sampled, inline with Fig. 6.

Limitations. The main drawback of GraVoS is the need of further training, which means longer training times than those of the original detectors. Furthermore, during training, the memory and the computational requirements are higher, due to the additional voxel selection stage. These limitations apply only for the training stage. At inference, the memory and the time are the same as in the original detector.

5. Conclusion

This paper has presented a novel and generic voxel selection method—*Gradient-based Voxel Selection (GraVoS)*. The key idea is to select voxels based on their meaningfulness to the detector. GraVoS was shown to address two fundamental challenges in 3D detection datasets, class-level data imbalance and foreground-background imbalance.

This paper has also proposed a training procedure that uses GraVoS to improve 3D detection without additional data. This is done by utilizing the selected voxels exclusively for fine tuning the detector.

We have demonstrated that training four SoTA voxel-based detectors using our training approach and selected voxels yields a boost in performance. The results are especially good when considering the challenging classes that have relatively few occurrences, regardless of difficulty.

An interesting future direction is to explore similar ideas of element selection on the raw cloud points. This will boost performance of detectors that do not use voxelization.

Acknowledgement. This work was supported by Advanced Defense Research Institute (ADRI)–Technion, Marie Sklodowska-Curie 893465, Israel Science Foundation 2329/22, and NVIDIA academic hardware grant.

References

[1] Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Andrew G Berneshawi, Huimin Ma, Sanja Fidler, and Raquel Urtasun. 3d object proposals for accurate object class detection. *Advances in neural information processing systems*, 28, 2015. 5

[2] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017. 5

[3] Yilun Chen, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Fast point r-cnn. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9775–9784, 2019. 2, 3, 5

[4] Shuyang Cheng, Zhaoqi Leng, Ekin Dogus Cubuk, Barret Zoph, Chunyan Bai, Jiquan Ngiam, Yang Song, Benjamin Caine, Vijay Vasudevan, Congcong Li, et al. Improving 3d object detection through progressive population based augmentation. In *European Conference on Computer Vision*, pages 279–294. Springer, 2020. 2, 3, 6

[5] Jaeseok Choi, Yeji Song, and Nojun Kwak. Part-aware data augmentation for 3d object detection in point cloud. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3391–3397. IEEE, 2021. 2, 3, 6

[6] Jiajun Deng, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li. Voxel r-cnn: Towards high performance voxel-based 3d object detection. *arXiv preprint arXiv:2012.15712*, 1(2):4, 2020. 2, 3, 4, 5, 6, 7, 8, 11, 12, 13

[7] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 2

[8] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012. 1, 2, 5

[9] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9224–9232, 2018. 2

[10] Benjamin Graham and Laurens van der Maaten. Submanifold sparse convolutional networks. *arXiv preprint arXiv:1706.01307*, 2017. 2

[11] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 43(12):4338–4364, 2020. 2

[12] Justin M Johnson and Taghi M Khoshgoftaar. Survey on deep learning with class imbalance. *Journal of Big Data*, 6(1):1–54, 2019. 2

[13] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3d proposal generation and object detection from view aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018. 5

[14] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019. 2, 5

[15] Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7345–7353, 2019. 5

[16] Liangchen Luo, Yuanhao Xiong, Yan Liu, and Xu Sun. Adaptive gradient methods with dynamic bound of learning rate. *arXiv preprint arXiv:1902.09843*, 2019. 12

[17] Rahul Mitra, Nitesh B Gundavarapu, Abhishek Sharma, and Arjun Jain. Multiview-consistent semi-supervised learning for 3d human pose estimation. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pages 6907–6916, 2020.

[18] Kemal Oksuz, Baris Can Cam, Sinan Kalkan, and Emre Akbas. Imbalance problems in object detection: A review. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3388–3415, 2020. 2

[19] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 918–927, 2018. 5

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

[21] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 3

[22] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *International conference on machine learning*, pages 4334–4343. PMLR, 2018. 3

[23] Matthias Reuse, Martin Simon, and Bernhard Sick. About the ambiguity of data augmentation for 3d object detection in autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 979–987, 2021. 2

[24] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10529–10538, 2020. 2, 3, 5

[25] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 770–779, 2019. 2, 3, 5

[26] Shaoshuai Shi, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. *IEEE transactions on pattern analysis and machine intelligence*, 43(8):2647–2664, 2020. 2, 3, 5, 6, 7, 8, 11, 12, 13

[27] Weijing Shi and Raj Rajkumar. Point-gnn: Graph neural network for 3d object detection in a point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1711–1719, 2020. 2

[28] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 761–769, 2016. 3

[29] Andrea Simonelli, Samuel Rota Bulo, Lorenzo Porzi, Manuel López-Antequera, and Peter Kontschieder. Disentangling monocular 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1991–1999, 2019. 6

[30] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 6, 11

[31] Kah-Kay Sung. Learning and example selection for object and pattern detection. 1996. 3

[32] OpenPCDet Development Team. Openpcdet: An open-source toolbox for 3d object detection from point clouds. https://github.com/open-mmlab/OpenPCDet, 2020. 8, 11, 12

[33] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, et al. Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, 23(9):661–692, 2006. 1

[34] Zhixin Wang and Kui Jia. Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1742–1749. IEEE, 2019. 5

[35] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 2, 3, 5, 6, 7, 8, 11, 12, 13

[36] Bin Yang, Ming Liang, and Raquel Urtasun. Hdnet: Exploiting hd maps for 3d object detection. In *Conference on Robot Learning*, pages 146–155. PMLR, 2018. 2, 5

[37] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7652–7660, 2018. 2, 5

[38] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11040–11048, 2020. 2, 3, 5

[39] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. IPOD: intensive point-based object detector for point cloud. *CoRR*, abs/1812.05276, 2018. 5

[40] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Std: Sparse-to-dense 3d object detector for point cloud. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1951–1960, 2019. 2, 3, 5

[41] Maosheng Ye, Shuangjie Xu, and Tongyi Cao. Hvnet: Hybrid voxel network for lidar based 3d object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1631–1640, 2020. 2, 3

[42] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11784–11793, 2021. 2, 5, 6, 7, 8, 11, 12, 13

[43] Wu Zheng, Weiliang Tang, Sijin Chen, Li Jiang, and Chi-Wing Fu. Cia-ssd: Confident iou-aware single-stage object detector from point cloud. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 3555–3562, 2021. 2

[44] Wu Zheng, Weiliang Tang, Li Jiang, and Chi-Wing Fu. Se-ssd: Self-ensembling single-stage object detector from point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14494–14503, 2021. 2

[45] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4490–4499, 2018. 2, 3, 5

6. Supplementary Material

In this paper we presented *GraVoS - Gradient-based Voxel Selection*, a novel and generic voxel selection method for voxel-based 3D object detection. To demonstrate the effectiveness of our approach we conduct a comprehensive study of various SoTA 3D detectors in Section 6.1. We also provide additional ablation studies in Section 6.2 and specific implementation details for each SoTA model in Section 6.3.

6.1. Additional SoTA Results

In the main paper we provided results of SoTA methods with and without our proposed data modification method. Here, we provide additional results for these methods.

Table 7 reports the results for the 3D detection benchmark and Table 8 reports the results for the Bird-Eye View (BEV) detection benchmark. The main additional result here is the error-reduction metric for each method on the different classes and difficulties. The error-reduction is specified by:

$$Error\ reduction = \frac{AP_{ours} - AP_{original}}{100 - AP_{original}} * 100,$$

where AP_{ours} represents our average-precision and $AP_{original}$ represents the original method's average-precision. We note that all the models were pre-trained using the original configuration [32] with batch size as mentioned in Section 6.3, to generate two training stage detectors: early $f(V;\theta_e)$ and late $f(V;\theta_l)$. The former was trained for only 1 epoch, while the latter was fully trained for 80 epochs.

Fig. 9 depicts some qualitative results of our proposed approach, for different object classes. It shows the gradients' magnitude for each voxel, as a color-map from blue to red, representing low to high values respectively, along with the final selected voxel subset. As can be seen, most of the objects' voxels have high gradient's magnitude value and therefore maintain theirs voxels after the selection stage. The background's voxels, however, usually do not have high gradient magnitude. Hence, less likely to be retained after the selection process.

Similar to the popular Dropout [30] augmentation, our data modification approach is general and can be applied to any voxel-based detection architecture. We showed that it is effective and can be used to improve multiple SoTA 3D detectors.

Table 9 provides comparison for the 3D detection benchmark. It shows comparison between the original detector, additional epochs training , and our voxel selection approach, for SECOND [35], Voxel R-CNN [6], and Part-A^2 [26] (configuration detailed in Section 6.3). Note that comparisons for CenterPoint [42] are not provided due to inconsistent results. We believe the inconsistencies are due to the fact that [42] was not originally designed and optimized for the KITTI dataset. Redesigning and optimizing detectors of previous works is beyond the scope of this work. The results show that in cases where a fully optimized detector is provided, GraVoS provides an improved detector.

6.2. Additional ablation studies

Interaction with different augmentations. We tested how our method interacts with different augmentations. Table 5 shows different augmentations protocols of [35] with and without our method. Table 5 shows that our method's gain is even higher when removing some augmentations such as GT sampling and Global augmentations i.e., rotation, flip, and scaling. Interestingly, when removing GT sampling and adding ours we achieve on-par performance as the baseline with GT sampling. This result further demonstrates the effectiveness of our approach, since GT sampling requires explicit GT annotations, whereas ours does not.

Method	Aug.		Average performance			
	Global	GT	*Ped.*	*Cyc.*	*Car*	All
[35]	x	x	49.96	69.51	**83.80**	67.76
+ Ours	x	x	**52.43**	**71.13**	82.89	**68.81**
[35]	x	-	46.42	51.92	80.34	59.56
+ Ours	x	-	**48.87**	**69.11**	**83.46**	**67.15**
[35]	-	-	33.37	37.33	69.02	46.57
+ Ours	-	-	**35.50**	**42.33**	**70.02**	**49.28**

Table 5. Without other augmentation, our method is even more beneficial. This is especially evident as it requires no GT labeling.

Deeper architecture. We compared our method with a deeper backbone of [35]. We duplicated layers in the backbone 2,4 and 8 times. Table 6 shows that our method provides performance gains over the original detector while a modified deeper architecture worsens the performance a bit. The degraded performance can be explained by overfitting. This shows that the benefit of our method is not attributed to a more complicated and deeper network.

Method	Average performance			
	Ped.	*Cyc.*	*Car*	All
[35]	49.96	69.51	**83.80**	67.76
[35] + Ours	**52.43**	**71.13**	82.89	**68.81**
[35] (x2)	50.11	68.23	83.17	67.17
[35] (x4)	50.00	69.82	82.80	67.54
[35] (x8)	48.14	65.97	82.71	65.61

Table 6. Our approach demonstrates improved performance over modified deeper architectures.

Voxel-selection alternatives. Align with Fig. 6 that shows alternative voxel selection methods on [35], we additionally provide results on [6]. Fig. 8 shows that while other alternatives barely improve the baseline, GraVoS greatly improve the baseline.

Method	*Car*			*Cyclist*			*Pedestrian*			Average			
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	*Car*	*Cyc.*	*Ped.*	All
SECOND [35]	**90.79**	**81.87**	**78.75**	81.85	65.42	61.26	54.90	49.84	45.15	**83.80**	69.51	49.96	67.76
Ours	89.53	81.06	78.07	**84.36**	**66.41**	**62.61**	**57.75**	**51.99**	**47.54**	82.89	**71.13**	**52.43**	**68.81**
Error reduction	-13.68	-4.47	-3.20	13.83	2.86	3.48	6.32	4.29	4.36	-5.62	5.31	4.94	3.26
Voxel R-CNN [6]	**92.62**	85.13	82.73	89.83	72.49	**68.87**	66.94	59.88	54.95	86.83	77.06	60.59	74.83
Ours	92.40	**85.41**	**82.84**	**91.97**	**72.98**	68.37	**68.52**	**61.63**	**56.71**	**86.88**	**77.77**	**62.29**	**75.65**
Error reduction	-2.98	1.88	0.64	21.04	1.78	-1.61	4.78	4.36	3.91	0.38	3.10	4.31	3.26
Part-A^2 [26]	**91.88**	**82.64**	80.21	89.45	71.71	67.74	65.37	58.43	53.62	84.91	76.30	59.14	73.45
Ours	91.68	82.58	**81.67**	**90.64**	**74.03**	**69.64**	**65.82**	**59.58**	**54.55**	**85.31**	**78.10**	**59.98**	**74.47**
Error reduction	-2.46	-0.35	7.38	11.28	8.20	5.89	1.30	2.77	2.01	2.65	7.59	2.06	3.84
CenterPoint [42]	**89.58**	**82.09**	**79.58**	80.27	62.85	60.13	56.85	53.17	49.73	**83.75**	67.75	53.25	68.25
Ours	88.74	81.74	79.53	**83.40**	**64.81**	**61.42**	**58.02**	**54.64**	**50.94**	83.34	**69.88**	**53.53**	**69.25**
Error reduction	-8.06	-1.95	-0.24	15.86	5.28	3.24	2.71	3.14	2.41	-2.52	6.60	0.60	3.15

Table 7. **Performance on the 3D detection benchmark.** Each method's performance is compared with and without our voxel selection. Results are reported for the Easy, Moderate (Mod.) and Hard categories on the three classes. Evidently, GraVoS improves the performance of all the methods for the non-prevalent classes, while it might slightly degrade the performance for the prevalent class. The average performance is always improved.

Method	*Car*			*Cyclist*			*Pedestrian*			Average			
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	*Car*	*Cyc.*	*Ped.*	All
SECOND [35]	92.30	**89.68**	**87.51**	87.87	**70.91**	66.57	60.94	55.73	51.56	89.83	75.12	56.08	73.67
Ours	**92.86**	89.62	87.26	**89.02**	70.88	**66.77**	**62.23**	**56.78**	**52.63**	**89.91**	**75.56**	**57.21**	**74.23**
Error reduction	7.27	-0.58	-2.00	9.48	-0.10	0.60	3.30	2.37	2.21	0.79	1.77	2.57	2.13
Voxel R-CNN [6]	**95.96**	91.43	**90.70**	93.63	76.09	**72.58**	69.97	63.60	59.04	**92.70**	80.77	64.20	79.22
Ours	**95.96**	**91.96**	89.49	**94.47**	**76.32**	71.60	**72.37**	**66.24**	**60.31**	92.47	**80.80**	**66.31**	**79.86**
Error reduction	0	6.18	-13.01	13.19	0.96	-3.57	7.99	7.25	3.10	-3.15	0.16	5.89	3.08
Part-A^2 [26]	**92.89**	**90.14**	**88.17**	91.19	75.42	70.97	68.31	61.70	57.33	**90.40**	79.19	62.45	77.35
Ours	92.85	90.07	88.13	**93.13**	**75.91**	**72.68**	**68.51**	**62.40**	**58.04**	90.35	**80.57**	**62.98**	**77.97**
Error reduction	-0.56	-0.71	-0.34	22.02	1.99	5.89	0.63	1.83	1.66	-0.52	6.63	1.41	2.74
CenterPoint [42]	**92.26**	**89.30**	**88.10**	83.84	66.40	63.05	61.26	58.08	54.83	**89.89**	71.10	58.06	73.01
Ours	91.91	88.90	88.00	**85.68**	**68.22**	**64.51**	**62.32**	**59.19**	**55.80**	89.60	**72.80**	**59.10**	**73.84**
Error reduction	-4.52	-3.74	-0.84	11.39	5.42	3.95	2.74	2.65	2.15	-2.87	5.88	2.48	3.08

Table 8. **Performance on the Bird Eye View (BEV) detection benchmark.** Similarly to Table 1, our method is beneficial for all four detectors.

6.3. Implementation details

This work was done using our reproduction of the pre-trained models provided by the publicly available OpenPCDet toolbox [32]. For a fair comparison, the default configurations for all the detectors were used for this reproduction. During fine-tuning with our voxel selection stage (GraVoS), the detectors were trained for additional epochs. When fine-tuning a deep neural networks it is usually required to change the Learning rate (LR) and Weight Decay (WD). The optimizer may also have effect on the fine-tuned network. Recent work [16] has shown that while Adam optimizer known to converge faster than *Stochastic Gradient Decent (SGD)*, it may have generalization degradation. This degradation may be caused by extreme learning rates that are usually used in the end of training. Hence, a new configuration for the optimizer is required.

To realize this idea, we subdivide the fine-tuning process into two steps. In the first step we fine-tune each detector for E_1 epochs using its original optimizer – *Adam-onecycle*. While for the second step we train for E_2 epochs using SGD with a step decay scheduler. This way we get the convergence speed from the first step while maintaining good generalization at the second step (end of training).

Specifically, the *Adam-onecycle* used in the first step is an Adam optimizer wrapped with Cosine-annealing scheduler that rises for part of the period and then declines. We set the rising time to be 30% of the optimizer total number of epochs E_1. In practice we chose $E_1 = 40$. The SGD optimizer, in the second step, includes a step decay scheduler. We set 2 steps for the step decay scheduler, S_1 and S_2. These steps, S_1 and S_2, have been chosen based on E_2, where at each step we reduced the learning rate by a factor of 10. In practice, we chose $S_1 = 7$ and $S_2 = 13$ for the steps and

Method	*Car* Easy	*Car* Mod.	*Car* Hard	*Cyclist* Easy	*Cyclist* Mod.	*Cyclist* Hard	*Pedestrian* Easy	*Pedestrian* Mod.	*Pedestrian* Hard	Average *Car*	Average *Cyc.*	Average *Ped.*	Average All
SECOND [35]	**90.79**	**81.87**	**78.75**	81.85	65.42	61.26	54.90	49.84	45.15	**83.80**	69.51	49.96	67.76
SECOND†	90.05	81.50	78.55	81.52	65.47	61.65	54.74	49.44	44.57	83.37	69.55	49.58	67.50
Ours	89.53	81.06	78.07	**84.36**	**66.41**	**62.61**	**57.75**	**51.99**	**47.54**	82.89	**71.13**	**52.43**	**68.81**
Voxel R-CNN [6]	92.62	85.13	82.73	89.83	72.49	68.87	66.94	59.88	54.95	86.83	77.06	60.59	74.83
Voxel R-CNN†	**92.69**	85.25	**82.85**	90.55	**73.06**	**69.62**	65.64	59.58	54.75	**86.93**	77.74	59.99	74.89
Ours	92.40	**85.41**	82.84	**91.97**	72.98	68.37	**68.52**	**61.63**	**56.71**	86.88	**77.77**	**62.29**	**75.65**
Part-A^2 [26]	91.88	82.64	80.21	89.45	71.71	67.74	65.37	58.43	53.62	84.91	76.30	59.14	73.45
Part-A^2†	**92.06**	**82.71**	**81.78**	88.86	72.86	68.53	**66.00**	58.71	53.86	**85.52**	76.75	59.52	73.93
Ours	91.68	82.58	81.67	**90.64**	**74.03**	**69.64**	65.82	**59.58**	**54.55**	85.31	**78.10**	**59.98**	**74.47**

Table 9. **Training scheme comparison on the 3D detection benchmark.** Each detector with † represents the detector after continuing to train with the same number of epochs, learning rate, and scheduler as in our method but without our voxel selection module.

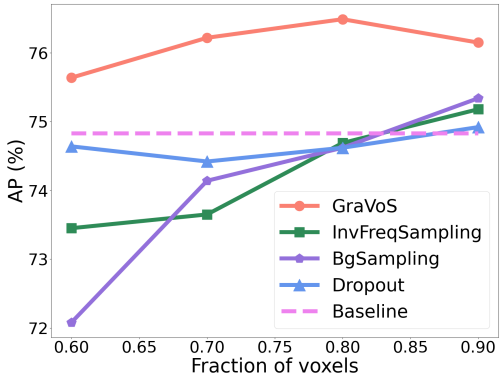


Figure 8. **Comparison to alternative approaches.** GraVoS is compared to Dropout, BgSampling and InvFreqSampling for different voxel selection ratios. All experiments were conducted on [6]. The baseline is the constant performance of the detector (all voxels). GraVoS outperforms other approaches significantly.

$E_2 = 20$ for the SGD optimizer total number of epochs.

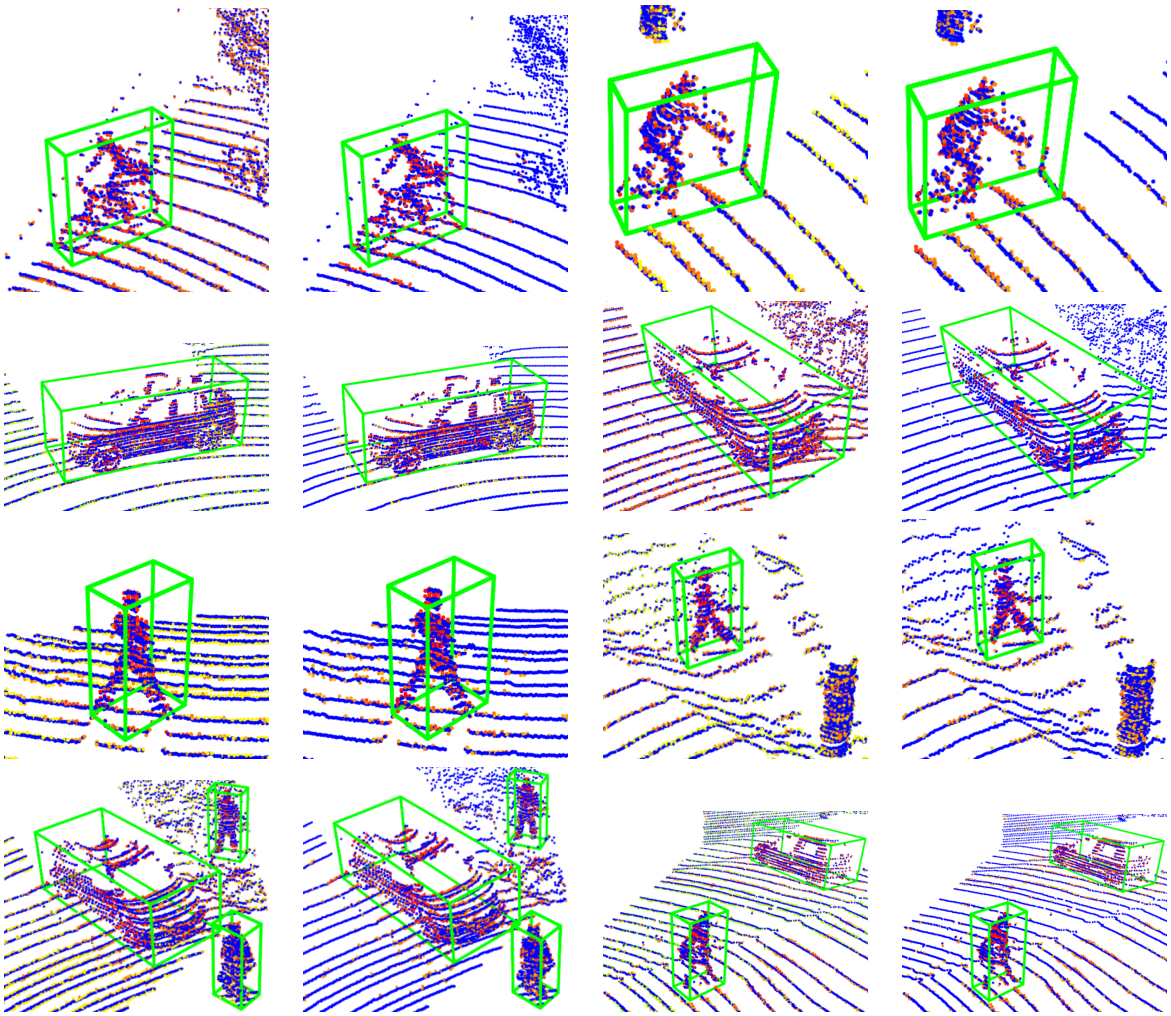
For each detector, the LR and WD were usually chosen to be about half of the original values. The specific configuration for each detector is given hereafter, where the voxel dimension are set to be (0.05, 0.05, 0.1) for all detectors.

SECOND [35]. For SECOND we set the batch size to be 4. The LR and WD of the first step are set to 0.005, whereas for the second step we changed the LR and WD to 0.003. Within our voxel selection stage (GraVoS), the location loss *rpn_loss_loc* was used.

Voxel R-CNN [6]. For Voxel R-CNN we set the batch size to be 4. The LR and Weight Decay (WD) of the first step are set to 0.005, whereas for the second step we changed the LR and WD to 0.003. Within our voxel selection stage (GraVoS), the location loss *rpn_loss_loc* was used.

Part-A^2 [26]. For Part-A^2 we set the batch size to be 4. The LR and WD of the first step are set to 0.005, whereas for the second step we changed the LR and WD to 0.003. For the voxel selection used in this detector, we chose the *rpn_loss* loss which composed of the box regression and the classifier losses.

CenterPoint [42]. For CenterPoint we set the batch size to be 4. The LR was set for the first and second steps to 0.002. The WD of the first step was set to 0.005, whereas for the second step we decreased it to 0.003. For the voxel selection used in this detector, we chose the *hm_loss_head_0* loss which corresponds to the *center heatmap head* presented in the original paper [42]. This loss is essentially equivalent to the location loss in other methods.



(a) Gradinent magnitudes (b) Final subset S^{mf} (a) Gradinent magnitudes (b) Final subset S^{mf}

Figure 9. **Gradient-based voxel selection visualization.** In each row we have two pairs of images (left pair and right pair). Each pair represents the gradient magnitude (a) along with the final choice subset gradient magnitudes (b). The top row depicts the *Cyclist* class, where in the second and third rows we have the *Car* and *Pedestrian* classes respectively. At the bottom row we have two sub-scenes with multiple classes. The magnitude of the gradients is depicted as a color-map from blue to red representing low to high values.