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.

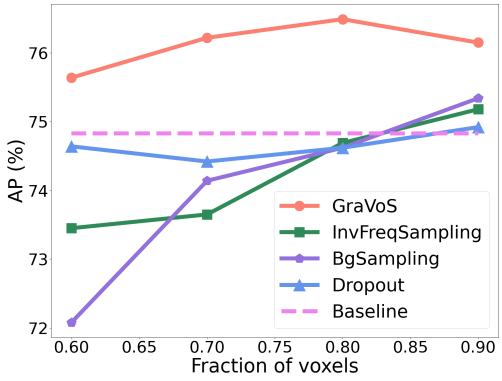


Figure 8. **Comparison to alternative approaches.** GraVoS is compared to Dropout, BgSampling and InvFreqSampling for different voxel selection ratios. All experiments were conducted on [6]. The baseline is the constant performance of the detector (all voxels). GraVoS outperforms other approaches significantly.

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

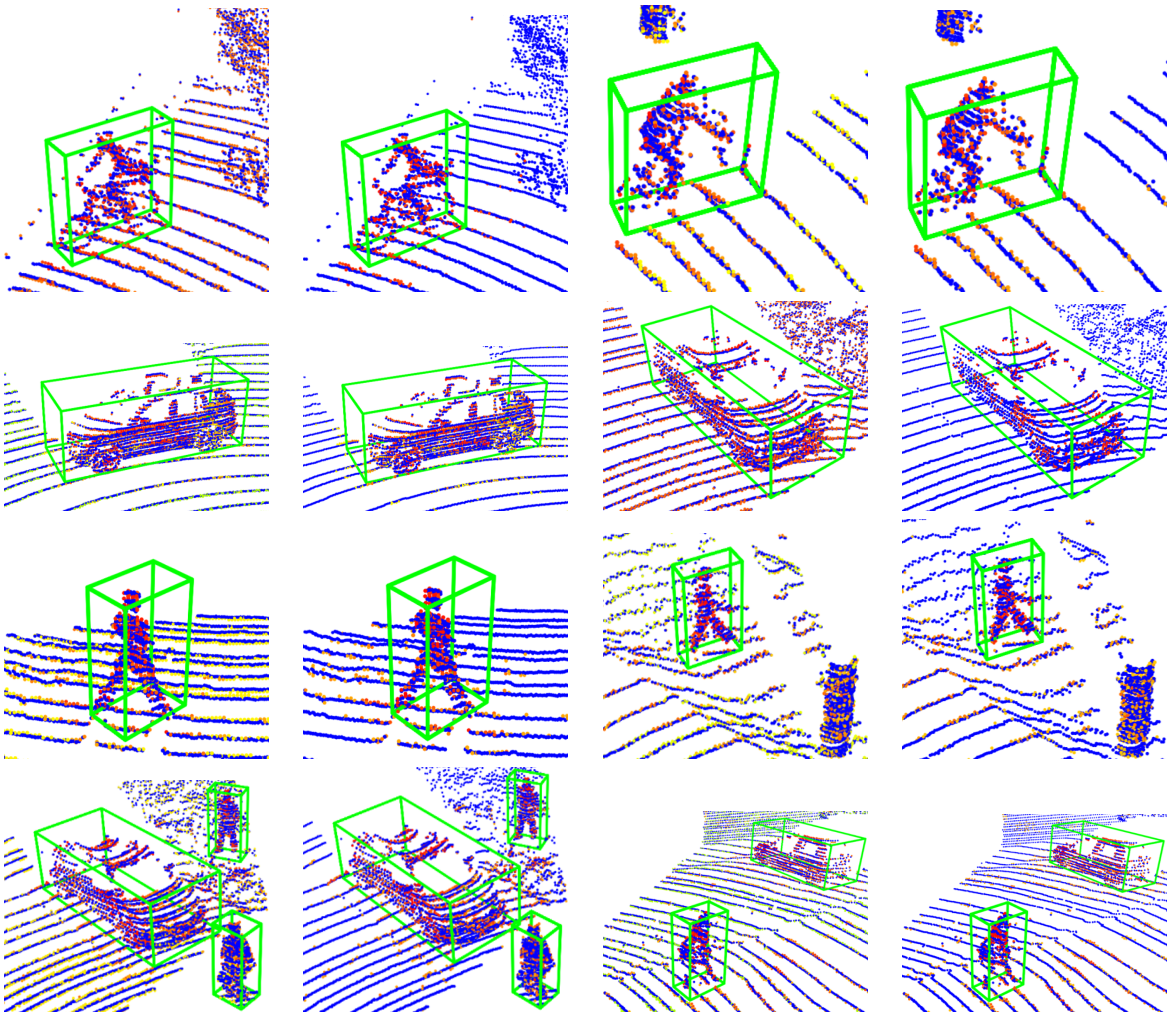
For each detector, the LR and WD were usually chosen to be about half of the original values. The specific configuration for each detector is given hereafter, where the voxel dimension are set to be (0.05, 0.05, 0.1) for all detectors.

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.