About the Ambiguity of Data Augmentation for 3D Object Detection in Autonomous Driving

Matthias Reuse, Martin Simon
Valeo Schalter und Sensoren GmbH
Hummendorfer Str. 76, Kronach, Germany
{matthias.reuse, martin.simon}@valeo.com

Prof. Dr. rer. nat. Bernhard Sick
Universität Kassel
Wilhelmshöher Allee 73, Kassel, Germany
bsick@uni-kassel.de

Abstract

Although data augmentation is considered an important step in the training strategy of 3D object detectors on point clouds to increase the overall performance and robustness, in almost all publications the topic of augmentation and the choice of the individual augmentation methods used are only addressed very briefly with reference to previous work and are not backed up with sufficient experiments. The question therefore arises as to the impact and the transferability of different augmentation policies. Through a series of elaborate experiments with four networks on two datasets, this paper shows that the positive effects of different data augmentation methods are not so clear-cut and instead depend strongly on the network architecture and the dataset.

1. Introduction

Over the course of the past years Deep Learning has been established as a powerful tool for solving many different tasks in various fields. It is well known by now, that the quality of the results heavily depends on the diversity and the amount of data the models are trained with. Since the recording and labelling of training data is in most cases very time consuming and costly, data augmentation methods were established to artificially increase the size and diversity of the training data. Especially in the field of autonomous driving, specifically the task of 3D object detection on point clouds, where a lot of labeled data is required, the precision of networks is a major safety factor. Since recording rare events is very difficult, augmentation comes in handy. Therefore, a variety of different augmentation methods ranging from very simple transformations to more sophisticated methods has been developed and widely used for this task. Even though in most publications the usage of these data augmentation methods is reported, the overall augmentation policy is in most cases not reasoned. Thus, in many papers the augmentation policy of previous works is simply carried over and not backed up with sufficient experiments. There are attempts to shed some light on the dubious field of augmenting 3D data for the task of object detection, but none has presented detailed experiments with a satisfying degree of generality yet [3, 8, 32]. Either the amount of different network architectures is not sufficient, the experiments are only performed on a single dataset or the augmentation policy remains occluded. Thus, neither a general statement about the effects of the individual augmentation methods nor the transferability of augmentation policies is possible.

Therefore, in this work a set of elaborate experiments regarding data augmentation for 3D object detection is performed to fill the aforementioned gap. In distinction from existing work these experiments are performed on different networks and datasets covering the current state of the art approaches of 3D object detection. In the course of this paper it is shown, that the effects of individual augmentation methods strongly depended on the network architecture as well as the underlying dataset and thus the transferability of augmentation policies, as it is current practice, is not ideal.

2. Related Work

For both 3D object detection and its data augmentation the current state of the art with respect to the present question of research is presented in the following section.

2.1. 3D Object Detection

In the current state of the art the research focuses mainly on two different groups of network architectures for 3D object detection distinguished by the amount of stages used. For the first group only one stage is utilized. This means only one network is used that produces the final box predictions directly. In contrast, the second group exploits a second stage. Here, in a first stage, no final results are created, but only proposals for box predictions. In a second stage, these proposals are further refined to the final predictions.

Most one stage detectors do not work directly on the continuous point clouds but first transform them into a discrete voxel grid. One of the first approaches used for 3D object detection was introduced in [13]. Here, the authors propose a network architecture consisting of several three dimensional convolutional layers. As input serves a point cloud mapped to a three dimensional occupancy voxel grid with handcrafted occupancy models. In [24] and [11] similar ideas are pursued and further developed still using handcrafted feature vectors for each voxel cell. For the approach proposed in [33] the feature vector for each voxel cell is created from the points falling inside one voxel by a small network inspired by [15]. Afterwards this voxel grid is converted by convolutional layers into a two dimensional feature map and further processed to achieve the box predictions. The authors of [26] continue the idea and apply sparse convolution layers and changes to the loss functions, but keep the general functionality. In [9] the working principle is further enhanced and the voxel grid is reduced to a grid of pillars. More recent works build further upon the backbones of [26] and [9] by implementing an anchor free detection head [2] or utilizing a multi scale approach [23]. In [32] a spatial-semantic feature aggregation module is introduced following the sparse convolutions to enhance the feature extraction capabilities.

So far all methods are voxelization approaches. There also exist one stage detectors working directly on the point cloud such as [28] where an architecture based on PointNet++ [16] for feature extraction is used, but they are less common. Thus, most of the object detectors taking the raw point cloud as input are two stage detectors. PointRCNN [18] is a two stage detector that also utilizes PointNet++ as a feature extraction followed by a proposal generation for the 3D boxes. These proposals are afterwards refined by a second network to the final box predictions. In [29] a two stage detector is presented, that is generating box proposals based on the raw point cloud with spherical anchors and for the second stage represents each proposal as a voxel grid. There exist also two stage detectors with voxelized point clouds as input such as [19]. Here, a encoder decoder network is applied to the voxel grid to create 3d box proposals. These are further refined in the second stage of the overall architecture by a simultaneous foreground segmentation and intra-object part location. In [17] an additional keypoint sampling is applied to further increase the box refinement. The authors of [5] propose another two stage voxelization approach exploiting a voxel RoI Pooling for the refinement of the proposals. In [30] the two stage voxelization approach is enhanced by using a center based representation of the box proposals.

2.2. Augmentation

Many different augmentation methods have been developed and established for the task of 3D Object detection on point clouds. Most commonly used are simple transformations of the whole point cloud and all groundtruth boxes accordingly. Thus, almost every work reports a global rotation of the point cloud [6, 9, 10, 12, 18, 19, 20, 23, 25, 26, 27, 29, 33]. Most of them also make use of a global scaling augmentation method. Here all points and groundtruth boxes are scaled by the same scaling factor in all directions. A global translation is not utilized as often. Only a few publications report its usage [9, 12, 20]. Another approach to globally augment the whole point cloud is flipping. In theory this can be performed along all three axes, in practice it is mostly utilized along the longitudinal axis.

The presence of labels for individual objects in the point cloud allows augmentations to be performed not only on the entire point cloud, but on each individual object by applying transformations to the bounding boxes of the objects and the points they contain. So many publications report such a local rotation or translation of individual objects [6, 9, 25, 26, 29, 33]. Note that local scaling of the point cloud is not used nearly as often although global scaling is commonly applied. In [4] another approach for local augmentation is presented called part aware augmentation. Here, the points belonging to an object are further split into parts of the object and individually augmented.

In addition to these basic transformations in [26] a groundtruth sampling approach is presented, where the individual objects and contained points are gathered from point clouds and randomly inserted into other clouds. This method was adopted by other works as well [9, 18, 23, 25, 29] and has become one of the common augmentation methods. In [34] an enhancement of this groundtruth sampling is mentioned regarding the placement of these new objects. Additionally, a categorization of the dataset according to the different classes is applied to tackle the problem of class imbalance of the nuScenes dataset [1].

Because of the success of the groundtruth sampling in the original paper, experiments with this method have been performed in some works, showing its capability to improve the networks detection quality [18, 32]. This often remains the only reported experiment regarding augmentation. In [32] and [31] small experiments are performed, but no detailed augmentation methods or their parameters are mentioned. More detailed experiments have been performed in [8]. Here, a series of experiments is carried out based on PointPillars [9] used for the detection of cars on the KITTI dataset [7]. Several augmentation methods are evaluated with various parameters. They note that some augmentation methods surprisingly lead to worse results and mention the importance of data augmentation at least for the KITTI dataset with its relatively small size and the PointPillars network. Experiments with other networks or datasets are not performed. In [3] a reinforcement learning approach is shown with the goal to learn the best augmentation policy for StarNet [14], which is a point based approach that tends more into the group of one stage detectors since there is no secondary refinement network, and PointPillars on the KITTI and Waymo [21] dataset. The authors use a variety of different, mostly global augmentation techniques and run their reinforcement learning algorithm. It is shown, that augmentation is able to improve the performance of the two networks and that their via reinforcement learning determined augmentation policy yields better results than a manually determined policy. But no detailed results are available and only vague indications are formulated regarding the single augmentation methods such as the underperformance of the horizontal flip method. The authors also introduce two new augmentation techniques using frustums. In the first method, called frustum dropout by the authors, a random point is selected and all points in the frustum around that point are randomly deleted. The second method, called frustum noise, does the same but adds points within the frustum instead of deleting them. The usage of both is not mentioned in any other publications.

3. Method

In this section the augmentation methods as well as the networks used for the experiments will be introduced.

3.1. Augmentation Methods

In the following let $p_i = (x_i, y_i, z_i)$ be the points of an arbitrary point cloud P and $b_j = (c^x_j, c^y_j, c^z_j, l_j, h_j, w_j, \phi_j)$ the elements of a set B of the according groundtruth boxes, where $c = (c^x, c^y, c^z)$ denotes the center position, l, h, w denote the dimensions and ϕ is the yaw angle of the boxes. Let P^* and B^* be the augmented sets of points and boxes accordingly. The parameters used are based on the current state of the art and are in most cases congruent to those that have proven to be the best in [8] for the respective augmentation method. All methods are illustrated in figure 1 for a minimal example.

3.1.1 Global Augmentation

Global augmentations refer to those methods, that are applied to the whole point cloud and all annotations in the same way [8]. Here, the groundtruth sampling is also referred to as global augmentation method, since it changes the scene in a global sense as well.

Groundtruth sampling adds additional objects to the current scene. These objects, i.e. the bounding box and its inner points, are previously collected in a database from the part of the dataset used for training. For each point cloud objects are randomly drawn from the database so that 15 objects are present at the current scene. Before an object is inserted into the current cloud a collision test to the already existing objects is performed. If a new object collides with an object already existing in the point cloud it gets discarded. Thus, at most $15 - |B|$ objects are added to one scene. Additionally, the database is filtered such that only objects with more than 5 points are included.

Global Flip mirrors the whole point cloud and all groundtruth boxes along the x axis if $q \in B(1, 0.5)$ equals 1 such that all points $p^*_i = (x_i, -y_i, z_i)$ and all boxes $b^*_j = (c^x_j, -c^y_j, c^z_j, l_j, h_j, w_j, \phi_j + \pi)$. A flip along other axes is also possible. Since for the Kitti dataset [7] the point cloud is cropped to the corresponding front camera image, it is not used in the experiments.

Global Translation shifts each point of the cloud by random values for each direction such that the augmented points $p^*_i = (x_i + \Delta x, y_i + \Delta y, z_i + \Delta z)$ and the centers of the augmented groundtruth boxes $c^*_j = (c^x_j + \Delta x, c^y_j + \Delta y, c^z_j + \Delta z)$ with $\Delta x, \Delta y, \Delta z \in N(0, 0.25)$ drawn for each point cloud from a Gaussian Distribution.

Global Rotation performs a rotation around the vertical z-axis for the whole point cloud and all objects accordingly in the way that all $p^*_i = R_z(\psi)p_i$ and all $c^*_j = R_z(\psi)c_j$ with $\psi \in U(-\frac{\pi}{4}, \frac{\pi}{4})$ drawn from an uniform distribution.

Global Scaling applies an isotropic scaling by a factor s in all three dimensions to the points and the boxes accordingly so $p^*_i = s \cdot (x_i, y_i, z_i)$ and $b^*_j = (s \cdot c^x_j, s \cdot c^y_j, s \cdot c^z_j, s \cdot l_j, s \cdot h_j, s \cdot w_j, \phi_j)$ with $s \in U(0.95, 1.05)$ drawn from an uniform distribution.

3.1.2 Local Augmentation

Local augmentations are not applied to the whole point cloud P but only to the groundtruth boxes B and the points within the bounding boxes. Thus, only the objects inside the point cloud like cars or pedestrians are effected by this augmentation techniques. Before an object is augmented a collision test is applied. There also exist variations of object augmentations, where not only the points within a box but also a portion of the surrounding points are augmented in the same way to keep more of the local information [32]. Though, for the following experiments the first presented approach is used because it is more often utilized so far. In the following, let P_{b_j} be the points enclosed by an arbitrary bounding box $b_j \in B$.

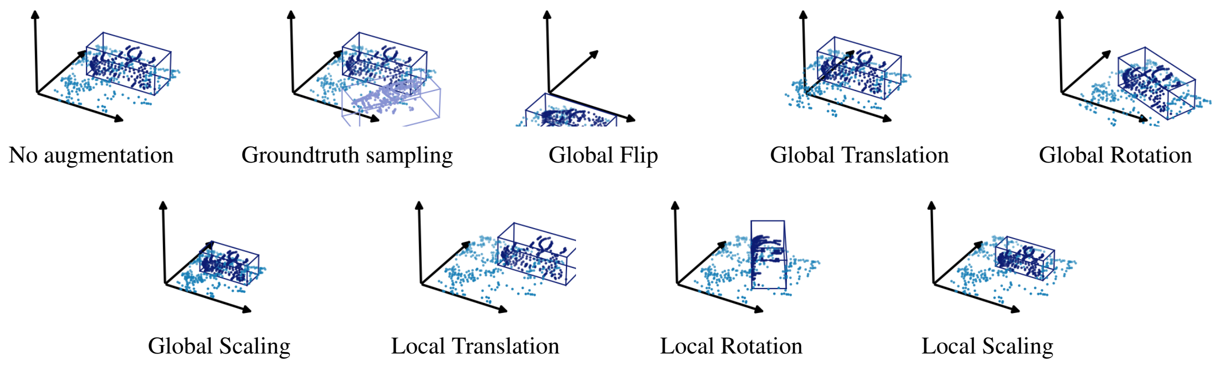

Figure 1: Illustration of the different augmentation methods for a minimal example.

Local Translation shifts each bounding box and its inner points by random values for each direction, such that $p^*_{ib_j} = (x_i + \Delta x_j, y_i + \Delta y_j, z_i + \Delta z_j)$ and $c^*_j = (c^x_j + \Delta x_j, c^y_j + \Delta y_j, c^z_j + \Delta z_j)$ if box b^*_j does not collide. The translation vector $(\Delta x_j, \Delta y_j, \Delta z_j) \in N_3(0, 0.25)$ is independently drawn for each bounding box from a Gaussian Distribution.

Local Rotation independently rotates each bounding box and its inner points by a random angle around the vertical axis of the center of the bounding box. Let $R^j_z(\psi_j)$ be the according rotation matrix, then $p^*_{ib_j} = R^j_z(\psi_j)(p_{ib_j} - c_j) + c_j$ and $\phi^*_j = \phi + \psi_j$ if box b^*_j does not collide. The rotation angles $\psi_j \in U(-\frac{\pi}{4}, \frac{\pi}{4})$ are independently drawn for each bounding box from an uniform distribution.

Local Scaling isotropically scales each bounding box and its inner points with respect to the center of the bounding box such that $p^*_{ib_j} = s_j \cdot (x_i, y_i, z_i)$ and $b^*_j = (c^x_j, c^y_j, c^z_j, s_j \cdot l_j, s_j \cdot h_j, s_j \cdot w_j, \phi_j)$ if box b^*_i does not collide. The scaling factor $s_j \in U(0.95, 1.05)$ is independently drawn for each bounding box from an uniform distribution.

3.2. Networks

For the experiments four networks were chosen according to the basic working principles of the current state of the art mentioned in section 2.

PointPillars is a one stage detector based on a variation of voxelization [9]. The inital point cloud is converted into a grid of pillars. For each pillar a feature encoding is performed. The resulting feature map is fed forward through a backbone consisting of two dimensional convolutions. The final predictions are performed by a SSD detection head.

PointRCNN is a two stage detector working directly on point clouds [18]. The first stage consists of a point cloud encoder decoder to create point-wise feature vectors. These are fed into two different modules. One is generating three dimensional boxes based on bins, the other is segmenting foreground points that actually belong to the object. The box proposals are enlarged by a constant factor and all points inside these regions of interest are extracted and used as input for the second stage together with the segmentation mask and the learned feature vector for each point of interest. Afterwards, the points belonging to each proposal are transformed into a canonical coordinate system with the center of the box proposal as the origin and a fixed heading angle along the longitudinal x-axis. These proposal are then refined to the final box predictions by applying an encoder network to the proposal points and the features extracted in the first stage.

PartA2 is a two stage detector as well but utilizing a voxelgrid as input [19]. The point cloud is first converted into such a discrete voxelgrid, where each voxel contains the average values of points falling inside. First, the voxel grid is convoluted to a feature map by an encoder. Following [26] this feature map is used as input for a RPN that generates the first box proposals. Additionally, the feature map is also deconvoluted to get voxelwise features. These are used to learn a foreground segmentation and an intra-object part location for each foreground point. For the second state the points within the regions of interest are transformed and normalized to canonical coordinates and converted into a voxelgrid. The intra-object part locations for the proposed points are further processed to match the max pooled features per point from stage one. Afterwards, both are concatenated and fed into a small sparse convolutional network, which outputs the final box refinement and scores for each box.

3DSSD is like PointPillars a single stage approach, but uses point clouds directly as input [28]. First, a backbone

Method	mAP Moderate Car ↑			
	PointPillars	PointRCNN	PartA2	3DSSD
no aug	58.72 0.00	78.24 0.00	75.43 0.00	61.51 0.00
full aug	**76.71** **17.99**	**78.73** **0.49**	**79.62** **4.18**	**78.69** **17.18**

Table 1: Results for Kitti moderate Car mAP on validation split with no augmentation and full augmentation for PointPillars, PointRCNN, ParA2 and 3DSSD. The best results for each network are marked in bold. Colored values are the difference to no augmentation.

network down samples each point cloud via farthest point sampling based on the spatial distance and feature distance to create global features for all representative points of the initial point cloud. Afterwards, points chosen via feature distance sampling are sampled and shifted towards the instance centers to create candidate points. These candidate points are then used to create further features. Finally, these set of features is used as input for an anchor free regression head.

4. Experiments

In the following section the performed experiments are presented. For the experiments the KITTI [7] dataset is used. As usual, the publicly available part of the dataset is split into a training set containing 3712 frames and a validation set with 3769 frames. The general experimental setup is oriented to [3] and [8]. Therefore, each training is performed and evaluated three times on the training and validation set respectively and the best results of all runs according to the mean average precision for the moderate car difficulty are reported. All experiments are trained and evaluated on the car class only to reduce the complexity of the results. The augmentation techniques are applied on the fly during the training in the order they are presented in section 3.1 if not stated otherwise. The implementation for the different networks as well as the augmentation methods is based on OpenPCDet [22]. PointPillar, PointRCNN and PartA2 are trained for 80 epochs in total, while 3DSSD is trained for 120 epochs as suggested by the OpenPCDet authors.

4.1. Experiment I - Impact of Augmentation

In the first experiment on the Kitti dataset all four presented networks in section 3.2 are trained completely without any augmentation and with all augmentation methods presented. Based on previous works [3, 8, 32] it is expected that the application of augmentation increases the quality of the detection results of the networks by a considerable margin. The results can be seen in table 1. For all four networks it can be seen, that full augmentation yields to better results than no augmentation. For PointPillars the increase of the mAP metric for moderate car is 17.99 percentage points. Thus, 23.45% of the networks performance stems from the augmentation, which is consistent with the results of [8]. The same can be observed for 3DSSD. Here, augmentation with every of the above mentioned methods increases the precision by 17.18 percentage points, which means 21.83% of the performance is due to the applied augmentation methods. So far, these results are in line with the expectations formulated above. The precision for PointRCNN increases as well with augmentation compared to without augmentation, But in contrast to PointPillars and 3DSSD the results for PointRCNN only improve by the very small margin of 0.49 percentage points, which is only 0.39% of the overall precision. A similar, though not as drastic, observation can be made with the PartA2 network. The precision of the network does increase by 4.18 percentage points, or 5.25% of the overall performance, but this is significantly less than when compared to PointPillars and 3DSSD. This may be because PointPillars and 3DSSD are one stage detectors, while PointRCNN and PartA2 utilize a second stage for box refinement. Like described in section 3.2 the second stage of PointRCNN and PartA2 takes the points belonging to each box proposal as input and transforms them into canonical coordinates. Thus, global as well as the proposed local changes of the scene by the augmentation methods have only a small effect on the input of the second stage. Furthermore, it can be assumed that the box proposals from the first stage, which serve as input for the second stage, vary and are not the same for every unaugmented point cloud. Both might be the reason augmentation shows little effects on these two stage detectors. Since each of PointPillars and 3DSSD as well as PointRCNN and PartA2 are a point based and voxelization approach, the input format of the point cloud does not seem to affect the effectiveness of the augmentation.

4.2. Experiment II - One at a time

In the previous experiment it was seen, that the impact of augmentation in general depends on the network architecture, mainly the existence of a second stage. The question remains if, even if the overall impact differs, the impact of the individual augmentation methods is consistent. Thus, for the second experiment each augmentation method is applied on its own and compared to training without any augmentation.

The results can be seen in table 2. At a first glance it is visible, that the different augmentation methods do not have the same effect for each network. Groundtruth sampling for example increases the results of PointPillars by 1.13 percentage points and thus has a positive impact on the training. For 3DSSD this positive impact is with an improvement of 14.22 percentage points of the mean average

Augmentation method	mAP Moderate Car ↑			
	PointPillars	PointRCNN	PartA2	3DSSD
no augmentation	58.72 0.00	78.24 0.00	75.43 0.00	61.51 0.00
+ groundtruth sampling	59.85 1.13	78.04 -0.20	76.00 0.57	**75.73** **14.22**
+ global flip	65.25 6.53	78.69 0.45	76.89 1.46	65.64 4.13
+ global translation	63.09 4.37	77.94 -0.30	78.14 2.71	61.78 0.27
+ global rotation	**67.28** **8.56**	78.26 0.02	**78.78** **3.35**	74.48 12.97
+ global scaling	63.34 4.62	**78.71** **0.47**	77.70 2.27	64.11 2.60
+ local translation	54.40 -4.32	77.70 -0.54	78.08 2.64	55.71 -5.80
+ local rotation	54.15 -4.57	77.99 -0.25	78.02 2.58	67.50 5.99
+ local scaling	61.07 2.35	78.52 0.28	76.82 1.39	63.61 2.10

Table 2: Results for Kitti moderate Car mAP on validation split with no augmentation and each augmentation method applied on its own. The best results for each network are marked in bold. Colored values are the difference to no augmentation.

precision disproportionately larger. Thus, the relative impacts are different but overall positive for both networks. Looking at groundtruth sampling in case of PointRCNN as well, it can be observed, that its usage actually has a negative impact of -0.20 percentage points for the PointRCNN network. For PartA2 groundtruth sampling yields an improvement of the results again, but only by a relatively small margin. Similar observations regarding the different effects of the single augmentation methods can be made for global translation as well. Here, again the positive impact of the augmentation differs for PointPillars, PartA2 and 3DSSD, while yielding worse results for PointRCNN. For local translation and local rotation the results are inconsistent as well across the four networks. While for PointPillars and PointRCNN the precision ends up lower, both augmentation methods cause better results for PartA2 and both better and worse results for 3DSSD respectively. Compared to the results reported in [8], that also performed experiments with the PointPillars network like stated in section 2, an inconsistency for PointPillars itself can be found with these two augmentation methods. The authors report a positive effect of local translation and rotation for the same set of parameters. This indicates, that not only the general architecture of the network but also the specific implementation is relevant. The other augmentation methods applied on its own yield better results for all four networks with changing influence. The best results are achieved by using global rotation for PointPillars and PartA2, global scaling for PointRCNN and groundtruth sampling for 3DSSD.

The high fluctuation of the results for each single augmentation method is rather counter intuitive and hard to explain. Neither the input format of the point cloud nor the existence of a refinement network seem to be decisive, since the results of the two groups are inconsistent in both cases. Despite the ambiguous results a few statements can be formulated. First, it seems that the global augmentation techniques are of more importance. They yield in almost all cases better results, in many cases of considerably large margin. The best results for each network are achieved by an augmentation method modifying the point cloud in a global manner. The local augmentation methods, that were used in this experiment, show on the other hand less consistent and as high improvements. Reason here might be the disturbance of local context and the physical properties of LiDAR scans. Thus, local translation leaves unnatural wholes in the point cloud and the rotation of single objects results in wrong points of view. Local scaling, on the other hand, changes the local context and physical properties only slightly and thus yields the best results of the local augmentation methods, although it has a comparatively small impact. Then again, the results of groundtruth sampling can be used to argue against this reasoning, as groundtruth sampling also disturbs the local context by adding objects such as cars off road for example. Once more, this represents an inconsistency, thus the influence of augmentation by and on the receptive field of the network remains unclear.

Method	mAP Moderate Car ↑			
	PointPillars	PointRCNN	PartA2	3DSSD
full aug	76.71 0.00	78.73 0.00	79.62 0.00	78.69 0.00
Only positive	**77.63** **0.92**	**78.80** **0.07**	**79.62** 0.00	**78.86** **0.17**

Table 3: Results for Kitti moderate Car mAP on validation split with positive augmentation methods based on table 2 for PointPillars, PointRCNN, PartA2 and 3DSSD. The best results for each network are marked in bold. Colored values are the difference to full augmentation.

Table 3 shows the results for all four networks trained with only the augmentation methods that yielded an improvement in the mean average precision for each network. It can be seen, that the precision for PointPillars, PointRCNN, and 3DSSD further improve by a small margin compared to the usage of all augmentation methods considered here. Note that for PartA2 all augmentations methods yielded an improvement and thus the precision is the same as for full augmentation. That the results improve if negative augmentations are omitted was to be expected. But the impact of leaving out these augmentations methods is rather small compared to the negative influence when applied on their own. While the usage of local translation and rotation worsens the results of PointPillars by more than four percentage points each, omitting these augmentation methods only leads to an improvement of 0.92 percentage points. A similar observation can be made for the other networks as well. The different augmentation methods overlap and the influence of each single augmentation techniques seems to be weakened as a result. Thus, applying full augmentation seems to be a valid policy, which yields good, although not optimal results, without the need of extensive experiments.

4.3. Experiment III - Leave one out

Augmentation method	mAP Moderate Car ↑			
	PointPillars	PointRCNN	PartA2	3DSSD
full Augmentation	76.71	78.73	79.18	78.69
	0.00	0.00	0.00	0.00
- groundtruth sampling	72.93	78.41	79.18	74.92
	-3.78	-0.32	-0.44	-3.77
- global flip	76.41	78.32	79.59	78.41
	-0.30	-0.41	-0.02	-0.28
- global translation	77.09	78.74	79.46	78.39
	0.38	0.01	-0.16	-0.30
- global rotation	76.68	78.50	**79.69**	78.30
	-0.03	-0.23	**0.08**	-0.39
- global scaling	76.79	78.58	79.63	78.62
	0.08	-0.15	0.02	-0.07
- local translation	**78.04**	78.67	79.50	**78.82**
	1.33	-0.06	-0.12	**0.13**
- local rotation	77.20	**78.86**	79.60	78.75
	0.49	**0.13**	-0.02	0.06
- local scaling	76.67	78.57	79.65	78.15
	-0.04	-0.16	0.04	-0.54

Table 4: Results for Kitti moderate car on validation split with full augmentation and full augmentation with one augmentation method left out respectively. The best results for each network are marked in bold. Colored values are the differences to full augmentation.

The results presented in the last section were ambiguous and hard to interpret. Thus, no general augmentation policy could be derived. To further investigate the ambiguity of the different augmentation methods another experiment is performed. This time the networks are trained with full augmentation with one augmentation method left out. It is to be expected, that the relative improvements or deterioration of the previous experiment can be also found when leaving out the according augmentation method. The results can be seen in table 4. In accordance with the previous results from table 2 the results when leaving out groundtruth sampling, global flip, global rotation and local scaling get worse for PointPillars, while the precision increases if local translation and local rotation are left out. But it can be observed, that the impact of the augmentation methods differs. In case of PointPillars for example the omission of groundtruth sampling decreases the precision by 3.78 percentage points, but increases the results only by 1.13 percentage points if used on its own. The same, but the other way around, can be observed with global rotation and global scaling. The usage of global rotation improves the results by 8.56 percentage points if used on its own, but worsens the results by only 0.03 percentage points if omitted. Same can be observed with the other networks for global scaling for PointRCNN, global translation for PartA2 or groundtruth sampling for 3DSSD to name only a few. Furthermore, for some augmentation methods the influence is not clear comparing table 2 and 4, but rather ambiguous. Taking PointPillars as example again the results for global translation and global scaling improve when they are omitted, while this is also the case when they are the only augmentation methods used. This is an unexpected behaviour in the continuity of the augmentation methods, that can also be observed for the other networks. Thus, groundtruth sampling and local translation for PointRCNN, global rotation, global scaling and local scaling for PartA2 and local rotation and local scaling for 3DSSD show the same inconsistency of the results compared to table 2. The best results are achieved by omitting local translation for PointPillars, local rotation for PointRCNN, global rotation for PartA2 and local translation for 3DSSD.

Similar to the previous experiment II the results are hard to interpret and rather ambiguous, but the findings of the previous experiments can be encouraged. Again, it is not possible to derive one general augmentation strategy for all networks. Input format and the amount of stages seem not be relevant for the impact of the single augmentation methods. Overall, the global augmentation methods seem once again more favorable. In addition to the previously observed overlap of different augmentation methods, it can be concluded that there are manifold interactions between the augmentation methods resulting in the ambiguous behaviour which was observed. Because of this interactions the impact of the augmentation methods can not be correctly measured by applying them or omitting them on their own. Thus, such greedy strategies for finding an optimal augmentation policy are not appropriate.

4.4. Experiment IV - Validation on nuScenes

All previous experiments were performed on the Kitti dataset, which is compared to other state of the art datasets rather small. Thus, to validate the findings and get a feeling for the impact of augmentation on larger and more diverse datasets as well, the experiments I to III are repeated on the nuScenes dataset. The common train-validation split is used with 28130 frames for the train set and 6019 for validation. Again, all experiments are performed on the car class only, to simplify the results. Additionally, due to the much longer training time these are only performed once for PointPillars. Therefore, the experiments are not as thorough, but anyways sufficient to investigate further inconsistencies of augmentation on different datasets. Intuitively, one would expect the influence of data augmentation with the nuScenes dataset to be smaller, as the dataset is larger and more diverse, but the trends of the individual augmentation methods remain the same as with Kitti.

Augmentation method	mAP Car ↑	
no augmentation	73.05	
	0.00	
full augmentation	79.29	
	6.24	
	+	-
groundtruth sampling	75.16	79.26
	2.11	-0.04
global flip	77.25	79.57
	4.20	0.28
global translation	78.06	78.96
	5.01	-0.33
global rotation	**80.71**	79.17
	7.66	-0.13
global scaling	77.17	79.17
	4.12	-0.12
local translation	77.28	80.18
	4.23	0.89
local rotation	75.54	**81.69**
	2.49	**2.39**
local scaling	72.33	79.41
	-0.72	0.11

Table 5: Results for nuScenes car class mAP on validation split for PointPillars with no augmentation, full augmentation, each augmentation method applied on its own (+) and full augmentation with one augmentation method left out respectively (-). The best results for each network are marked in bold. Colored values are the differences to no augmentation for (+) and full augmentation for (-).

The upper half of table 5 shows the results of a training without and with all augmentation methods presented in section 3.1. It can be seen, that with augmentation a better mean average precision is reached, but the impact is much smaller as it was with Kitti for PointPillars. So now, only 7.87% of the network performance stem from augmentation instead of the previously reported $23, 45\%$. This behaviour was to be expected. The size of the nuScenes dataset and the amount of objects belonging to the car class in comparison to Kitti are most likely the reason for the observed effect.

In the lower half of table 5 the results for the other two performed experiments are shown, where the single augmentation methods are applied on its own or omitted from the overall set of augmentations respectively. Looking at the left-hand side, all global augmentation methods increase the mean average precision. Again, global rotation yields the best result, with 80.71 percentage points even better than the results for all augmentation methods at once. Interestingly, the increase for each global augmentation method is quite similar to that found for Kitti, while the overall increase for full augmentation is rather small. It is possible that the PointPillars model has reached its upper limit at around 80 percentage points, so that the potential of all augmentation methods cannot be fully developed. While the global augmentation methods show the expected behaviour, the local augmentation methods show inconsistencies to the results on Kitti. Local translation and local rotation increase the performance on nuScenes, while on Kitti the results were decreased. The same, but the other way around, can be observed for local scaling, although with the relative small decrease of -0.72 percentage points which may be due to the missing repetitions of the experiment and the resulting larger influence of the random effects during training.

Looking at the leave one out experiments also reported in table 5 incongruities similar to the experiments on the Kitti dataset can be found. While the global augmentation methods applied on its own increase the results by a considerably margin, omitting these decreases the results by a much smaller value. For example, groundtruth sampling applied on its own increases the results by 2.11 percentage points, if left out, the results only decrease by -0.04 percentage points. Furthermore, the ambiguity of some methods can be observed as well on the nuScenes dataset for global flip and global rotation, where omission as well as usage of the same augmentation methods increase the results. This was also obversed for PointPillars on the Kitti dataset but for different augmentation methods. Thus, other than intuitively expected the trends of the individual augmentation methods change with the usage of the nuScenes dataset. This might be due to a possible different distribution of the object parameters in the dataset, which again would not explain the inconsistencies for the experiments performed only on nuScenes. Therefore, it can be concluded that the effects of augmentation methods diverge not only for different network architectures but also for different datasets.

5. Conclusion

In this work a series of elaborate experiments was performed to show the impact of various augmentation techniques on 3D object detectors. It was shown, that the influence of augmentation on the results are ambiguous and hard to interpret. The two stage detectors were less influenced by augmentation, thus they generalize better than one stage detectors. In general there is no optimal augmentation policy for all networks, but for each network individually depending on the architecture of the network and the underlying dataset. Finding this individual optimal augmentation strategy is cumbersome, because of the overlap and manifold interactions between the different augmentation methods and the large search space. It was shown that using all of the most common methods presented in this work may not be ideal, but seems to be a good first attempt to test the effects of augmentation in general for the network at hand and to determine whether further efforts to identify the optimal strategy are worthwhile.

References

[1] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019. 2

[2] Q. Chen, L. Sun, Z. Wang, K. Jia, and A. Yuille. Object as hotspots: An anchor-free 3d object detection approach via firing of hotspots. In *ECCV*, 2020. 2

[3] S. Cheng, Z. Leng, E. D. Cubuk, B. Zoph, C. Bai, J. Ngiam, Y. Song, B. Caine, V. Vasudevan, C. Li, Q. V. Le, J. Shlens, and D. Anguelov. Improving 3d object detection through progressive population based augmentation. *ArXiv*, abs/2004.00831, 2020. 1, 3, 5

[4] J.-S. Choi, Y. Song, and N. Kwak. Part-aware data augmentation for 3d object detection in point cloud. *ArXiv*, abs/2007.13373, 2020. 2

[5] J. Deng, S. Shi, P. Li, W. Zhou, Y. Zhang, and H. Li. Voxel r-cnn: Towards high performance voxel-based 3d object detection. *ArXiv*, abs/2012.15712, 2020. 2

[6] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1355–1361, 2016. 2

[7] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 2, 3, 5

[8] M. Hahner, D. Dai, A. Liniger, and L. Gool. Quantifying data augmentation for lidar based 3d object detection. *ArXiv*, abs/2004.01643, 2020. 1, 2, 3, 5, 6

[9] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. Pointpillars: Fast encoders for object detection from point clouds. *CoRR*, abs/1812.05784, 2018. 2, 4

[10] J. M. Lehner, A. Mitterecker, T. A. Adler, M. Hofmarcher, B. Nessler, and S. Hochreiter. Patch refinement - localized 3d object detection. *ArXiv*, abs/1910.04093, 2019. 2

[11] B. Li. 3d fully convolutional network for vehicle detection in point cloud. *CoRR*, abs/1611.08069, 2016. 2

[12] M. Liang, B. Yang, S. Wang, and R. Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *The European Conference on Computer Vision (ECCV)*, September 2018. 2

[13] D. Maturana and S. Scherer. VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. In *IROS*, 2015. 2

[14] J. Ngiam, Benjamin Caine, W. Han, B. Yang, Y. Chai, P. Sun, Y. Zhou, Xi Yi, O. Alsharif, Patrick Nguyen, Z. Chen, Jonathon Shlens, and V. Vasudevan. Starnet: Targeted computation for object detection in point clouds. *ArXiv*, abs/1908.11069, 2019. 3

[15] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016. 2

[16] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *CoRR*, abs/1706.02413, 2017. 2

[17] S.i Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10526–10535, 2020. 2

[18] S. Shi, X. Wang, and H. Li. Pointrcnn: 3d object proposal generation and detection from point cloud. *CoRR*, abs/1812.04244, 2018. 2, 4

[19] S. Shi, Z. Wang, X. Wang, and H. Li. Part-a^2 net: 3d part-aware and aggregation neural network for object detection from point cloud. *CoRR*, abs/1907.03670, 2019. 2, 4

[20] K. Shin and M. Tomizuka. Improving a quality of 3d object detection by spatial transformation mechanism. *ArXiv*, abs/1910.04853, 2019. 2

[21] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. C. Y. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z.-F. Chen, and D. Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. *ArXiv*, abs/1912.04838, 2019. 3

[22] OpenPCDet Development Team. Openpcdet: An open-source toolbox for 3d object detection from point clouds. `https://github.com/open-mmlab/OpenPCDet`, 2020. 5

[23] B. Wang, J. An, and J. Cao. Voxel-fpn: multi-scale voxel feature aggregation in 3d object detection from point clouds. *CoRR*, abs/1907.05286, 2019. 2

[24] D. Wang and I. Posner. Voting for voting in online point cloud object detection. In *Robotics: Science and Systems*, 2015. 2

[25] Z. Wang, H. Fu, L. Wang, L. Xiao, and B. Dai. Scnet: Subdivision coding network for object detection based on 3d point cloud. *IEEE Access*, 7:120449–120462, 2019. 2

[26] Y. Yan, Y. Mao, and B. Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 2, 4

[27] B. Yang, W. Luo, and R. Urtasun. Pixor: Real-time 3d object detection from point clouds. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7652–7660, 2018. 2

[28] Z. Yang, Y. Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11037–11045, 2020. 2, 4

[29] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia. STD: sparse-to-dense 3d object detector for point cloud. *CoRR*, abs/1907.10471, 2019. 2

[30] T. Yin, X. Zhou, and P. Krähenbühl. Center-based 3d object detection and tracking. *ArXiv*, abs/2006.11275, 2020. 2

[31] W. Zhang, Z. Wang, and C. C. Loy. Multi-modality cut and paste for 3d object detection. *CoRR*, abs/2012.12741, 2020. 2

[32] W. Zheng, W. Tang, S. Chen, L. Jiang, and C.-W. Fu. Cia-ssd: Confident iou-aware single-stage object detector from point cloud. *ArXiv*, abs/2012.03015, 2020. 1, 2, 3, 5

[33] Y. Zhou and O. Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. *CoRR*, abs/1711.06396, 2017. 2

[34] B. Zhu, Z. Jiang, X. Zhou, Z. Li, and G. Yu. Class-balanced grouping and sampling for point cloud 3d object detection. *ArXiv*, abs/1908.09492, 2019. 2