IMPROVING DATA AUGMENTATION FOR MULTI-MODALITY 3D OBJECT DETECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Single-modality object detectors have witnessed a drastic boost in the past few years thanks to the well-explored data augmentation and training techniques. On the contrary, multi-modality detectors adopt relatively simple data augmentation due to difficulty in ensuring cross modality consistency between point clouds and images. Such a limitation hampers fusion effectiveness and performance growth of multi-modality detectors. Therefore, we contribute a pipeline, named transformation flow, to bridge the gap between single and multi-modality data augmentation with transformation reversing and replaying. In addition, considering occlusions, a point in different modalities may be occupied by different objects, making augmentations such as cut and paste non-trivial for multi-modality detection. We further present Multi-mOdality Cut and pAste (MoCa), which simultaneously considers occlusion and physical plausibility to maintain the multi-modality consistency. Without using ensemble of detectors, our multi-modality detector achieves new state-of-the-art performance on nuScenes dataset and competitive performance on KITTI 3D benchmark. Code and models will be released.

1 INTRODUCTION

Three-dimensional (3D) object detection is an essential vision task with wide applications such as in robotics and autonomous driving cars. In the context of autonomous driving, encouraging results (Shi et al., 2020) have been obtained with point cloud data from Light Detection and Ranging (LiDAR) devices. Nonetheless, these single-modality LiDAR-based methods also have limitations: a typical LiDAR system can only perceive objects in a limited range (Caesar et al., 2019), and cannot distinguish categories of similar structures, *e.g.*, pedestrians and trees.

Meanwhile, imagery features, by nature, play a complementary role. It is believed that imagery features can facilitate more accurate detection by providing richer semantic information. Much effort has geared towards adding RGB camera images to complement point cloud data (Cho et al., 2014; Liang et al., 2019; 2018; Qi et al., 2017b; Vora et al., 2020). However, compared to single modality detectors (e.g., 2D detectors (Lin et al., 2017; Ren et al., 2015) and LiDAR-based 3D object detectors (Shi et al., 2020; Yan et al., 2018; Yang et al., 2019)) with extensively explored single modality data augmentation (Yan et al., 2018; Yun et al., 2019; Fang et al., 2019) and training practices in the past few years, relevant techniques for multi-modality 3D detectors are under-explored and appear less aggressive than those in single-modality detectors, making the performance of multi-modality algorithms under expectation (Liang et al., 2018; Qi et al., 2017b).

Previous multi-modality detectors (Liang et al., 2019; 2018; Qi et al., 2017b) refrain from using rich data augmentations. This phenomenon is mainly due to the *difficulties in maintaining the consistency* between point cloud and images. In other words, multi-modality augmentations should ensure exact correspondence between a point in one modality and that of another modality after a series of transformations. With such a constraint, previous methods (Liang et al., 2018; Qi et al., 2017b) find difficulties in applying random flipping or rotations. An alternative way for finding the correspondence is to use the inputs before augmentation. However, it is not applicable to those points generated during model inference, *e.g.*, box coordinates or votes (Qi et al., 2019a; 2020).

To tackle this problem, we contribute a pipeline, named *multi-modality transformation flow*, to ensure multi-modality consistency to cope with rich set of augmentations. Multi-modality consistency can be maintained if and only if the augmentation can be reversed and replayed. Thus, *transforma-*

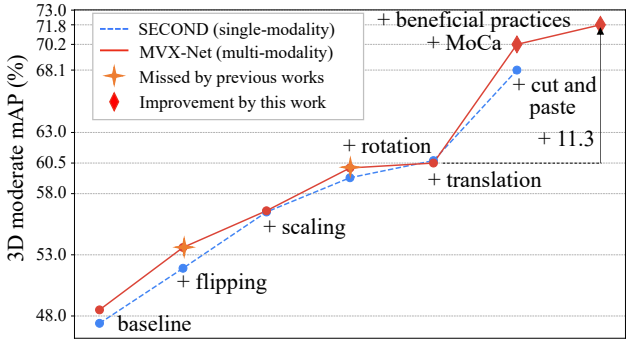


Figure 1: Step-by-step performance improvements brought by multi-modality augmentations, MoCa, and beneficial practices. Random flipping, scaling, rotation, and translation enabled by the *transformation flow* are equally effective for both SECOND and MVX-Net. *Multi-modality cut and paste (MoCa)* with beneficial practices further improves the performance of MVX-Net, surpassing its single-modality counterpart (SECOND with cut and paste) by a large margin on KITTI dataset.

tion flow records the parameters and orders of transformations used by each augmentation. Then in the multi-modality fusion process, any point in the LiDAR coordinates could find its corresponding image pixel coordinates by reversing the point cloud transformations and replaying the image transformation. In this pipeline, any transformation can be applied as long as it is invertible; thus, one can now apply augmentations commonly found in single-modality detectors with equal effectiveness for multi-modality detectors (Fig. 1).

Based on *multi-modality transformation flow*, we further propose a new augmentation approach, *multi-modality cut and paste (MoCa)*. State-of-the-art LiDAR-based detectors (Shi et al., 2020) benefit significantly from cut and paste augmentation (Yan et al., 2018). However, such an effective augmentation scheme is absent from multi-modality methods (Sindagi et al., 2019; Vora et al., 2020), which severely limits their performance. Single-modality cut and paste methods do not need to consider occlusions and physical plausibility in other modalities. Consequently, they may construct scenes containing objects completely occluded in 2D images or paste an object at implausible locations. These cases hinder the learning of multi-modality fusion modules as it makes the point from one modality fusing features from different objects in another modality. MoCa constrains paste operations to avoid occlusion between objects in both the bird's eye view (BEV) and the 2D imagery domain. To further improve the detector's generalizability, we use randomly selected intersection over foreground (IoF) to determine the occlusion in imagery domain. MoCa improves object detection performance by a large margin and it is readily applicable to many existing multi-modality detectors (Sindagi et al., 2019; Vora et al., 2020; Liang et al., 2018; Qi et al., 2018).

The transformation flow, MoCa, together with some beneficial practices, improve the mAP of MVX-Net (Sindagi et al., 2019), a strong and generic multi-modality detector, by **11.3%** moderate mAP on KITTI dataset (Fig.1) and **5.8%** mAP on nuScenes dataset. Without using an ensemble of class-specialized detectors, the enhanced MVX-Net achieves new state-of-the-art results on nuScenes dataset and obtains competitive results on KITTI 3D benchmark.

2 Methodology

We are curious with the limited performance gain after extending single-modality to multi-modality 3D object detectors. We find that multi-modality detectors use relatively fewer kinds of data augmentations than single-modality detectors do, due to the missing principle of maintaining multi-modality consistency in augmentation. To this end, we first contribute a pipeline, named *transformation flow*, to allow any invertible augmentations to be applied in multi-modality detection (Sec. 2.1). Then we propose *multi-modality cut and paste (MoCa)* to further improve the performance (Sec. 2.2). We choose MVX-Net (Sindagi et al., 2019) as a strong and generic baseline for our study. Finally, we explore different architecture design and training strategies for better performance (Sec. 2.3). The findings can be easily extended to other multi-modality 3D object detectors.

2.1 Transformation Flow

Data augmentation plays a pivotal role in improving the models generalizability. However, the augmentation strategies adopted by single-modality 3D detectors are more aggressive than those used by multi-modality methods. For example, global rotation and random flip are widely applied

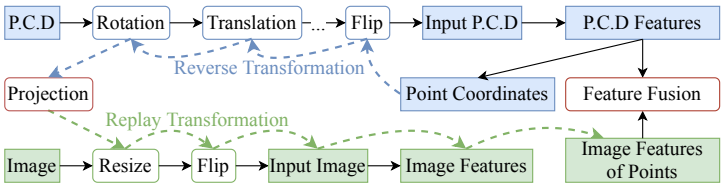


Figure 2: **Multi-modality transformation flow.** A point in point cloud data (P.C.D) finds its image coordinate and features by reversing point cloud augmentation and replaying image augmentation.

by single-modality methods (Shi et al., 2019; Yang et al., 2019; Liu et al., 2020) but are absent from some multi-modality methods (Sindagi et al., 2019; Liang et al., 2018; 2019) due to the *difficulties in maintaining the consistency* between point cloud and image data.

Multi-modality consistency. The essence of maintaining multi-modality *consistency* during augmentation is to maintain the correct correspondence between points and image pixels; thus, the points can still be correctly fused with their corresponding image features after a series of augmentations. While previous studies (Sindagi et al., 2019; Liang et al., 2018) have applied augmentation in multi-modality detection, they use much fewer augmentations than those single-modality methods and the issue of maintaining consistency across different modalities has not been systematically studied. Here, we contribute a pipeline named *multi-modality transformation flow*, which is useful for maintaining the multi-modality consistency during augmentation, enabling more aggressive augmentation strategies for multi-modality detection.

Multi-modality transformation flow. As shown in Fig. 2, the multi-modality transformation flow records all the transformations of point cloud and image data during data augmentations. Such transformation flow is required to transform the augmented data back for finding the correct correspondence between the point cloud and image pixels during fusion. Most augmentations are reversible, *i.e.*, they contain a forward transformation to augment the data, with a reverse transformation to transform the data back into its original state. Before training, the image and point cloud data are augmented by different augmentations independently. Note that the transformations of points are equivalent to transforming the LiDAR sensor to a new position, resulting in new point coordinates but not affecting the captured image as the camera is not transformed. Thus, rotation and translation can be applied to point cloud data, but they do not need to be applied to the image simultaneously.

Reverse and replay. With transformation flow, a point in one modality can obtain its corresponding point in another modality by reversing the augmentations of its own modality and replaying the augmentation of the other modality (Fig. 2). Specifically, during fusion, the reverse transformation of each augmentation is applied to the augmented points following the inverse order of point cloud augmentations (reverse). Next, we can safely project the points onto the imagery pixel coordinates using the calibration information of data (Geiger et al., 2013; Caesar et al., 2019). The projected points then obtain their corresponding image features after going through the forward transformations following the same order of the corresponding image augmentations (replay).

Applications. With *multi-modality transformation flow*, any augmentation, as long as it is reversible, can be used to augment multi-modality data without sacrificing the consistency. Therefore, now we can revisit and validate the gain of existing augmentation techniques that can be extended from single-modality to multi-modality augmentations. We compare these augmentation techniques step by step (Fig. 1); such experiments are new in the literature. The results show that global flipping, scaling, rotation, and translation are all essential augmentations to improve a multi-modality detector, enabled by the transformation flow. Meanwhile, this formulation is general and applicable to any kind of points of a modality such as the original points in the point cloud data (Vora et al., 2020), center of generated voxels (Sindagi et al., 2019), and predicted votes (Qi et al., 2020).

Some methods (Yan et al., 2018) apply a small amount of noise (*e.g.*, random translation and rotation) to each ground truth object separately. They can also be applied in *multi-modality transformation flow* by recording the transformation of each single ground truth objects. However, recent works suggest that such strategy either does not add value (Shi et al., 2020) or hurts the performance in some scenarios (Lang et al., 2019).

2.2 Multi-modality Cut and Paste

Cut and paste is effective to create a diverse combination of scenes and objects when data is limited (Dvornik et al., 2018; Fang et al., 2019; Dwibedi et al., 2017). It is a common augmentation

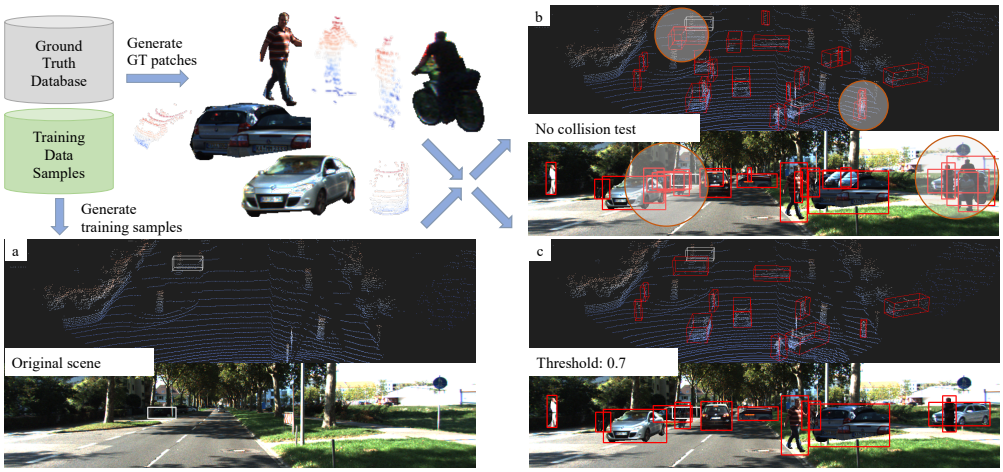


Figure 3: **Multi-modality cut and paste.** Given one training frame and some GT objects generated by the database, MoCa checks collisions in both BEV and 2D image based on Intersection over Foreground (IoF). It then pastes the valid patches of objects to all the modalities. A blind pasting operation will cause many heavily overlapped objects or put objects at implausible positions (circled area in (b)), which is far from the natural distribution of data. The gray and red bounding boxes indicate the original object in the current frame and the pasted objects, respectively. The original frame (a) contains very few objects and is enriched with more objects after multi-modality cut and paste (c). The figure is best seen in color.

technique in single-modality detectors but is absent in multi-modality detectors. Recently, SECOND (Yan et al., 2018) introduces cut and paste into the point cloud domain, named as ground truth sampling (GT-sampling). It is shown that GT-sampling not only accelerates model convergence but also reduces class imbalance issues. Therefore, almost all state-of-the-art single-modality 3D detectors (Shi et al., 2020; Yang et al., 2020) adopt GT-sampling to boost their performance.

It is non-trivial to consistently extend GT-sampling to the imagery domain in multi-modality methods. A point cloud patch that is visible from the bird's eye view (BEV) does not guarantee its corresponding image patch is also perceivable in the imagery domain. Often, an object may be occluded in the image plane and thus its imagery content only captures the features of the occluding object. A blind cut and paste would risk having inconsistent point cloud and imagery patches. To circumvent this intricate problem, multi-modality methods (Liang et al., 2019; Sindagi et al., 2019) resort to a lower starting baseline without using GT-sampling.

To our knowledge, this is the first work that investigates underlying challenges in multi-modality cut and paste for 3D object detection. We further propose a more general form of GT-sampling named *multi-modality cut and paste (MoCa)*. It is readily applicable to existing multi-modality methods (Sindagi et al., 2019; Vora et al., 2020; Liang et al., 2018; Qi et al., 2018) and brings substantial improvement.

MoCa first builds a ground truth database for each annotated object offline. Specifically, point cloud for each object and its corresponding image patch is cropped before training, using ground truth 3D bounding boxes and 2D masks, respectively. MoCa randomly samples point cloud-image patch pairs and paste them to the original scene according to their 3D bounding boxes and 2D masks (Fig. 3). The enriched scene will then go through the multi-modality augmentation process within the transformation flow and then will be used to trained the model. To avoid boundary artifacts caused by image patches, we follow (Dwibedi et al., 2017) to apply random blending to smoothen the boundaries of image patches. Such an operation is not needed for point cloud since the data is sparse. Notably, though it might take hours to build the GT database if the dataset is large, the cost of the training speed brought by MoCa is less than 5% due to the parallel data loaders.

Occlusion handling. The non-trivial part of multi-modality cut and paste lies in occlusion handling. Image-based cut and paste (Dwibedi et al., 2017; Dvornik et al., 2018) usually pastes objects at different locations in the image and ignores the physical plausibility. On the other hand, point cloud cut and paste (Yan et al., 2018) only avoids occlusion in BEV because objects are generally assumed to be on the same ground plane and well separated in BEV. Potential occlusions in 2D image are

neglected because current 3D object detectors (Yan et al., 2018; Yang et al., 2019; Lang et al., 2019) usually predict bounding boxes only from BEV. However, during the multi-modality fusion process, due to the occlusion, the projected points of an occluded object might obtain the image features of occluding objects (Fig. 3 (b)). This makes the image features ambiguous and increases the difficulty in training feature extractors. Therefore, not handling the occlusion in 2D image will affect the overall performance as validated by our experiments.

MoCa considers the consistency in both point cloud and image modalities. Specifically, given a batch of objects with their point cloud and their corresponding image patches, the multi-modality cut and paste first discards overlapped objects in BEV and then carefully handles the occlusion in 2D images. Given a set $P = \{p_i \mid i = 1, 2, ..., N\}$ containing the original objects and the objects to be pasted, we use Intersection-over-Foreground (IoF) of the objects' bounding boxes to represent the occlusion degree of an object p_i in the 2D image as

$$IoF(p_i, P) = \max\{\frac{p_i \bigcap p_j}{p_i} \mid j \neq i\}. \tag{1}$$

Once a sampled object's IoF is greater than a given threshold or that object makes any one of the original boxes' IoF greater than the given threshold, the sampled object will not be pasted in the current training iteration. The original objects will not be discarded.

Mixed IoF thresholds. Different IoF thresholds lead to a different number of objects being pasted. Therefore, we propose to use a mix of occlusion thresholds to provide more diverse occlusion cases and scenes during training to improve the detector's robustness and generalization ability. Specifically, given a threshold set (we use $\{0, 0.3, 0.5, 0.7\}$ in this work), a threshold will be randomly chosen from the set. Then objects whose occlusion degrees (in the batch) are greater than the threshold will be discarded during each iteration. The remaining objects' point cloud and image patches will then be pasted to the positions specified by their 3D and 2D bounding boxes *without random perturbation* to ensure consistency, respectively. Image patches are pasted in the order of their depth, *i.e.*, the farther the object, the earlier it is pasted.

2.3 Exploring Beneficial Practices

In this paper, we use MVX-Net (Sindagi et al., 2019) to study multi-modality augmentation as it is simple and generic. MVX-Net uses a pre-trained Faster R-CNN as an image feature extractor and adopts VoxelNets for 3D detection. For VoxelNet we adopt PointPillars (Lang et al., 2019) and SECOND (Yan et al., 2018) because they are efficient and generic, and are widely deployed in autonomous driving systems. During implementation, we explore some beneficial practices in architecture design and optimization.

Aligned pyramid feature fusion. For feature fusion, the point cloud is first transformed from LiDAR coordinates to camera coordinates and then projected to the image pixel coordinates[1]. Once the pixel coordinates P_{img} are obtained, MVX-Net selects image features using the quantized coordinates P'_{img} and concatenate them with the point cloud feature. However, as shown in Fig. 4, the quantization introduces feature misalignment in the fusion process since the quantized coordinates are not an accurate projection from the given points. Such misalignment brings adverse effects in multi-modality detection because the quantization is applied to at least 10K projected points with their corresponding features. Inspired by RoIAlign (He et al., 2017), we introduce *aligned feature fusion*, which uses differentiable bilinear sampling kernel (Jaderberg et al., 2015) to overcome the misalignment issue and obtain the feature of a given point as follows:

$$\sum_{H}^{n} \sum_{W}^{m} U_{nm} \max(0, 1 - \mid P_y - n \mid) \max(0, 1 - \mid P_x - m \mid), \tag{2}$$

where H and W are the height and width, respectively, of the feature map U, and $P_{img} = (P_x, P_y)$ is the pixel coordinates of a projected point.

We further enhance the MVX-Net (Sindagi et al., 2019) by adding a FPN (Lin et al., 2017) in image branch (Fig. 4). For the feature map in each scale, the pixel coordinate P_{img} is divided by the strides of the current feature map and then used to select the corresponding image features. Then we obtain

[1]Details are provided in the Section C.

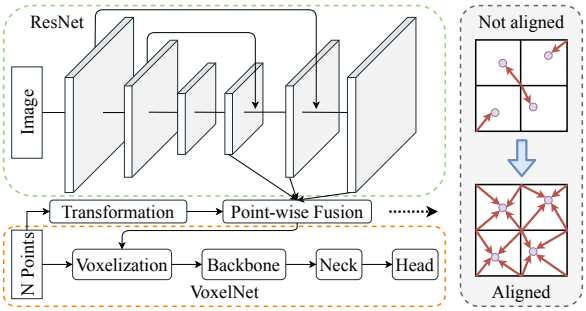


Figure 4: Enhanced MVX-Net. We fuse FPN features with point cloud features using not-quantized coordinate.

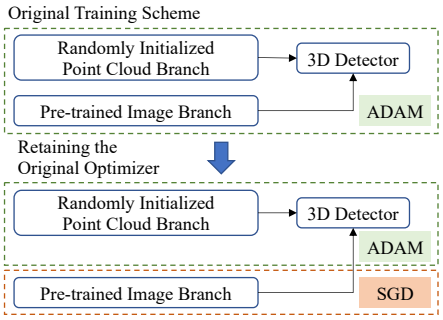


Figure 5: Retaining the original optimizer for each modality.

the multi-scale imagery features of each point from different scales of feature maps, concatenate them together, and fuse them with the points' point cloud features by a linear layer.

Retaining the original optimizers. Multi-modality 3D object detectors (Liang et al., 2018; Qi et al., 2018; Sindagi et al., 2019; Liang et al., 2019) usually use a pre-trained model for image feature extraction because there are limited image data in the problem domain. The image feature extractor is typically pre-trained on 2D recognition tasks (Russakovsky et al., 2015; Lin et al., 2014) using an SGD optimizer to ensure good performance. On the other hand, ADAM is shown superior in handling irregular and unstructured point cloud data (Qi et al., 2017a;b). Hence, for multi-modality 3D object detection, a common practice is to train the point cloud feature extractor from scratch jointly with the pre-trained image feature extractor using an ADAM optimizer. Empirically, we observe that the switch of optimizer from SGD to ADAM in the image branch causes a slight performance drop. We ameliorate this problem by retaining the original optimizer of each modality (Fig. 5). Specifically, we retain the use of a SGD optimizer for the image feature extractor and an ADAM optimizer for the point cloud branch during the joint training.

3 Experiments

We validate our framework on the KITTI dataset and nuScenes dataset.

KITTI dataset. The KITTI dataset (Geiger et al., 2012) contains 7481 training images and 7518 test images, both with their corresponding point cloud. We train all the models using the train split containing 3712 samples and evaluate them on the validation split consisting of 3769 samples, following previous works (Chen et al., 2017; Yan et al., 2018; Shi et al., 2019). The KITTI benchmark evaluates the models by Average Precision (AP) of each class (car, pedestrian, and cyclist) under easy, moderate, and hard conditions. For simplicity, we use mean AP over three classes to measure overall performance of the models in the ablation study.

nuScenes dataset. The nuScenes dataset (Caesar et al., 2019) contains 28130 synchronized multi-view images and point cloud samples for training, 6019 samples for validation, and 6008 samples for test benchmark. The dataset contains 10 categories and we evaluate the models on these 10 classes by NDS metric (Caesar et al., 2019). The NDS metric not only measures the mean AP but also takes translation, scale, orientation, velocity, and attribute errors of the true positives into considerations.

3.1 Ablation Study

Multi-modality transformation flow. We validate the effectiveness of *multi-modality transformation flow* with different multi-modality augmentation techniques. To obtain a clear comparison with SECOND (Yan et al., 2018), we conduct experiments on the vanilla MVX-Net (Sindagi et al., 2019) without using the beneficial techniques discussed in the previous section.

The original MVX-Net only uses global scaling in the augmentation, and global rotation and translation are thought as not applicable. *Multi-modality transformation flow* enables random flipping, translation, and rotation to be applied for multi-modality detectors. Table 1 verifies that random flipping, scaling, rotating, and translating point cloud are essential for both single-modality and

Table 1: Comparison of different augmentation strategies for single-modality 3D detector SECOND and multi-modality 3D detector MVX-Net on KITTI. The augmentations marked by † are absent in previous works and enabled by transformation flow. The bolded augmentations are proposed in this work. The augmentation technique is in order, whereby, each augmentation is added onto the previous ones sequentially

Method	Augmentation	mAP (%)	Pedestrian			Cyclist			Car		
		Mod.	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
SECOND	no augmentation	47.4	46.5	40.0	34.7	55.3	38.7	32.8	71.5	63.4	63.4
	+ flipping	51.9	57.6	50.9	45.1	53.7	37.6	37.2	77.3	67.3	65.8
	+ scaling	56.5	54.8	47.0	44.5	67.2	48.4	42.3	84.5	74.1	67.6
	+ rotation	59.3	59.8	53.0	46.6	66.5	48.8	48.0	87.1	76.0	68.9
	+ translation	60.7	61.0	54.6	51.9	69.0	50.7	49.2	87.2	76.8	75.6
	+ cut and paste	68.1	68.1	61.1	54.0	79.4	65.8	60.8	87.5	77.4	75.5
MVXNet	no augmentation	48.5	49.1	45.5	40.5	53.2	35.4	31.2	75.3	64.5	57.8
	+ flipping†	53.6	49.7	43.3	41.7	57.2	45.6	40.4	81.5	71.9	65.9
	+ scaling	56.6	54.4	47.0	44.6	66.2	49.5	42.9	82.7	73.4	67.1
	+ rotation†	60.1	62.7	54.7	52.8	67.2	49.5	42.6	87.7	76.0	68.8
	+ translation	60.5	62.7	55.6	53.2	65.7	49.1	48.4	87.2	76.9	75.5
	+ cut and paste	69.0	68.2	**62.3**	**56.7**	83.7	67.3	63.2	87.9	77.4	73.6
	+ MoCa (ours)	**70.2**	**68.6**	61.9	54.7	**86.0**	**71.2**	**65.0**	**87.9**	**77.6**	**76.0**

Table 2: Ablation study of each component. Modifications are added sequentially

(a) On KITTI validation set

Method	3D mAP (%)		
	Easy	Mod.	Hard
SECOND	78.3	68.1	63.4
+ image branch	71.9	60.5	59.1
+ MoCa	80.9	70.2	65.2
+ retain original optimizers	81.2	71.0	65.7
+ aligned pyramid fusion	**81.4**	**71.8**	**69.5**

(b) On nuScenes validation set

Method	NDS (%)	mAP (%)
PointPillars + FPN	53.4	40.1
+ image branch	54.6	41.8
+ MoCa	55.0	43.0
+ retain original optimizers	57.7	47.3
+ aligned pyramid fusion	**58.1**	**47.9**

multi-modality 3D detectors. With these augmentations added sequentially, both the single-modality (SECOND) and the multi-modality (MVX-Net) detectors obtain significant improvement.

Notably, the results also show that *multi-modality cut and paste* plays a critical role in bridging the performance gap between the multi-modality detector and single-modality detector. And the proposed MoCa improves the vanilla cut and paster by 1.2 moderate mAP. Both single-modality and multi-modality methods obtain large improvement with the help of cut and paste (moderate mAP from 60.7% and 60.5% to 68.1% and 70.2%, respectively). Without cut and paste, MVX-Net cannot even surpass SECOND (60.5% vs. 60.7%). But with the help of *MoCa*, MVX-Net surpasses its counterpart by a large margin (70.2% vs. 68.1%). More ablation studies of multi-modality cut and paste are in the appendix.

Step-by-step results on KITTI dataset. We evaluate the beneficial components step by step in Table 2a. We freeze image branch pre-trained on COCO and KITTI dataset when retaining the original optimizers. MVX-Net enhanced by our method significantly surpasses its previous version (second row in Table 2a) by **9.5%**, **11.3%**, and **10.4%** in mAP of easy, moderate, and hard conditions, respectively. The enhanced MVX-Net also surpasses its single-modality counterpart, SECOND, by **3.1%**, **3.7%**, and **6.1%** in mAP of easy, moderate, and hard conditions, respectively.

Step-by-step results on nuScenes dataset. We also validate the beneficial components step by step on the more challenging nuScenes dataset (Table 2b). Since existing results in the literature are mainly based on PointPillars, we also adopt PointPillars as the 3D detector in MVX-Net. We reimplement PointPillars but supplement it with FPN, which achieves 53.4% in the NDS score, higher than that reported by the dataset provider (44.2% (Caesar et al., 2019)). As shown in Table 2b, MoCa and the explored practices are all beneficial on nuScenes dataset. The proposed changes allow MVX-Net to surpass PointPillars by a large margin (7.8% mAP and 4.7% NDS).

3.2 Benchmark Results

KITTI dataset. We compare our method with other published methods on the KITTI 3D detection benchmark for completeness. Note that many previous works (Yang et al., 2019; Yan et al., 2018; Lang et al., 2019; Ku et al., 2018; Wang & Jia, 2019; Liu et al., 2020) train **specialized models with different hyperparameters for different categories and ensemble their results on the benchmark**. However, using multiple detectors for multiple classes is not ideal for real-world ap-

Table 3: Comparison with published multi-modality methods on KITTI 3D test benchmark. 'Ensembled' indicates whether the results are ensembled by class-specialized detectors. The best results are bolded

Method	Ensembled	mAP (%)			Pedestrian			Cyclist			Car		
		Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
AVOD-FPN	√	65.8	54.9	49.9	50.5	42.3	39.0	63.8	50.6	44.9	83.1	71.8	65.7
F-PointNet		68.3	56.0	49.2	50.5	42.2	38.1	72.3	56.1	49.0	82.2	69.8	60.6
PointPainting		70.0	58.8	53.6	50.3	41.0	37.9	77.6	63.8	55.9	82.1	71.7	67.0
F-ConvNet		**73.8**	**61.6**	54.0	**52.2**	43.4	38.8	**82.0**	**65.1**	**56.5**	**87.4**	**76.4**	66.7
MoCa + MVX-Net++ (ours)	×	71.0	60.2	**54.7**	50.9	**43.7**	**40.0**	76.1	61.0	53.4	86.0	75.9	**70.7**

Table 4: Comparison with previous methods on nuScenes validation set. 'Con. Veh.', 'Ped.', and 'T.C.' are the abbreviations of construction vehicle, pedestrian, and traffic cone, respectively. 'FA' means FreeAnchor and '3×' means longer training schedule. NDS score, mAP, and APs of each categories are reported. The single class AP not reported in the paper is marked by '-'. The best results are bolded

Method	Modality	NDS	mAP	Car	Truck	Bus	Trailer	Con. Veh.	Ped.	Motor	Bicycle	T.C.	Barrier
PointPillars	L	46.8	28.2	75.5	31.6	44.9	23.7	4.0	49.6	14.6	0.4	8.0	30.0
3D-CVF	L + I	49.8	42.2	79.7	37.9	55.0	36.3	-	71.3	37.2	-	40.8	47.1
PointPillars+FPN	L	53.4	40.1	80.6	35.9	43.5	29.2	5.4	71.9	34.9	11.8	35	52.6
3DSSD	L	56.4	42.6	81.2	47.2	61.4	30.5	12.6	70.2	36.0	8.6	31.1	47.9
PointPainting	L + I	58.1	46.4	77.9	35.8	36.1	37.3	15.8	73.3	41.5	24.1	62.4	60.2
CenterPoint	L	65.0	56.6	84.6	54.7	66	32.3	15.1	84.5	56.9	38.6	67.4	66.1
MoCa + MVX-Net++	L+I	58.1	47.9	82.4	41.5	49.6	28.6	9.1	79.1	50.3	27.2	49	61.9
MoCa + MVX-Net++ on stronger PointPillars baselines													
+ FA	L + I	60.3	52.9	83.6	48.5	56.4	31.4	10.8	81.6	61.0	35.7	58.1	61.7
+ FA + RegNetX-400MF		62.1	55.2	84.2	51.7	63.6	34.2	18.0	82.0	61.8	32.9	59.4	64.0
+ FA + RegNetX-1.6GF		64.7	58.4	85.2	55.5	64.5	35.0	20.8	84.5	68.0	42.8	62.3	65.1
+ FA + RegNetX-3.2GF		65.4	59.2	85.7	56.8	66.2	35.8	21.7	84.4	67.4	44.2	62.8	66.3
+ FA + RegNetX-3.2GF + 3×		**67.8**	**62.5**	**87.0**	**60.8**	**68.2**	**39.7**	**23.9**	**85.8**	**70.2**	**51.6**	**68.1**	**69.5**

plications. Therefore, in this work, we train all our models on all three classes without tuning the models for specific categories. Table 3 shows that MoCa achieves promising performance among multi-modality methods. Despite using a single model for all three classes, MoCa achieves competitive performance against other baselines that use an ensemble of class-specific detectors. On hard conditions, only MoCa obtains top-3 results of all the categories. In comparison, other methods do not exhibit such generalizability.

nuScenes dataset. We compare our method with other published methods on the validation set of nuScenes dataset (Caesar et al., 2019) in Table 4. The dataset has more diverse scenes and need to detect 10 categories, which is more challenging. We report the performance of enhanced MVX-Net with MoCa, using the image branch from HTC pre-trained on nuImages dataset. The results show that MoCa surpasses PointPainting (Vora et al., 2020), the previous multi-modality state of the art, by **1.5%** mAP. To evaluate the scalability of our methods and compare with methods using large model (Zhu et al., 2019), we also report the performance of MoCa based on stronger baselines enhanced by FreeAnchor (Zhang et al., 2019), RegNetX (Radosavovic et al., 2020), and longer schedule with stronger augmentations. Our method brings consistent improvement over strong single-modality baselines. MoCa achieves new state-of-the-art results not only on the overall metric, but also on the AP of all the categories. The final performance of MoCa surparsses the previous best result achieved by a large model of CenterPoint (Yin et al., 2020) trained by CBGS (Zhu et al., 2019), with an absolute improvement of **2.8%** NDS and **5.9%** mAP.

4 Conclusion

This paper investigates and discusses the pitfalls of applying data augmentations to multi-modality 3D object detection. We contribute a pipeline, *multi-modality transformation flow*, to ensure consistency during augmentations and to enable a richer set of augmentation strategies. Based on that, we validate the effectiveness of different augmentations that are absent in previous works and further present *multi-modality cut and paste (MoCa)* to bridge the gap of augmentations between multi-modality and single-modality 3D detectors. Under different strong baselines, our method improves the performance consistently and achieves new state-of-the-art performance on nuScenes dataset.

REFERENCES

Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. *CoRR*, abs/1903.11027, 2019.

Zhaowei Cai and Nuno Vasconcelos. Cascade R-CNN: Delving into high quality object detection. In *CVPR*, 2018.

Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. Hybrid task cascade for instance segmentation. In *CVPR*, 2019a.

Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019b.

Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3D object detection network for autonomous driving. In *CVPR*, 2017.

Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Huimin Ma, Sanja Fidler, and Raquel Urtasun. 3d object proposals using stereo imagery for accurate object class detection. *TPAMI*, 2018.

Yilun Chen, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Fast Point R-CNN. In *ICCV*, 2019c.

Hyunggi Cho, Young-Woo Seo, B. V. K. Vijaya Kumar, and Ragunathan Rajkumar. A multi-sensor fusion system for moving object detection and tracking in urban driving environments. In *ICRA*, 2014.

MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. `https://github.com/open-mmlab/mmsegmentation`, 2020.

Jiajun Deng, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li. Voxel R-CNN: towards high performance voxel-based 3d object detection. *CoRR*, abs/2012.15712, 2020.

Terrance Devries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout. *CoRR*, abs/1708.04552, 2017.

Nikita Dvornik, Julien Mairal, and Cordelia Schmid. Modeling visual context is key to augmenting object detection datasets. In *ECCV*, 2018.

Debidatta Dwibedi, Ishan Misra, and Martial Hebert. Cut, paste and learn: Surprisingly easy synthesis for instance detection. In *ICCV*, 2017.

Lue Fan, Xuan Xiong, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Rangedet: In defense of range view for lidar-based 3d object detection. *CoRR*, abs/2103.10039, 2021.

Hao-Shu Fang, Jianhua Sun, Runzhong Wang, Minghao Gou, Yong-Lu Li, and Cewu Lu. InstaBoost: Boosting instance segmentation via probability map guided copy-pasting. In *ICCV*, 2019.

Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012.

Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *I. J. Robotics Res.*, 2013.

Georgios Georgakis, Arsalan Mousavian, Alexander C. Berg, and Jana Kosecka. Synthesizing training data for object detection in indoor scenes. In *RSS*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *ICCV*, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *NeurIPS*, 2015.

Nitish Shirish Keskar and Richard Socher. Improving generalization performance by switching from adam to SGD. *CoRR*, abs/1712.07628, 2017.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L. Waslander. Joint 3D proposal generation and object detection from view aggregation. In *IROS*, 2018.

Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. PointPillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019.

Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *ECCV*, 2018.

Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-task multi-sensor fusion for 3D object detection. In *CVPR*, 2019.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014.

Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *CVPR*, 2018.

Zhe Liu, Xin Zhao, Tengteng Huang, Ruolan Hu, Yu Zhou, and Xiang Bai. TANet: Robust 3D object detection from point clouds with triple attention. In *AAAI*, 2020.

Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with warm restarts. In *ICLR*, 2017.

Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum PointNets for 3D object detection from RGB-D data. In *CVPR*, 2018.

Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3D object detection in point clouds. In *ICCV*, 2019a.

Charles R. Qi, Xinlei Chen, Or Litany, and Leonidas J. Guibas. ImVoteNet: Boosting 3d object detection in point clouds with image votes. In *CVPR*, 2020.

Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *CVPR*, 2017a.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017b.

Lu Qi, Li Jiang, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Amodal instance segmentation with kins dataset. In *CVPR*, 2019b.

Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollar. Designing network design spaces. In *CVPR*, 2020.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015.

Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. PointRCNN: 3D object proposal generation and detection from point cloud. In *CVPR*, 2019.

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 *CVPR*, 2020.

Vishwanath A. Sindagi, Yin Zhou, and Oncel Tuzel. MVXNet: Multimodal voxelnet for 3D object detection. In *ICRA*. IEEE, 2019.

Sourabh Vora, Alex H. Lang, Bassam Helou, and Oscar Beijbom. PointPainting: Sequential fusion for 3D object detection. In *CVPR*, 2020.

Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark E. Campbell, and Kilian Q. Weinberger. Pseudo-lidar from visual depth estimation: Bridging the gap in 3D object detection for autonomous driving. In *CVPR*, 2019.

Zhixin Wang and Kui Jia. Frustum ConvNet: Sliding frustums to aggregate local point-wise features for amodal 3D object detection. In *IROS*, 2019.

Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

Yan Yan, Yuxing Mao, and Bo Li. SECOND: Sparsely embedded convolutional detection. *Sensors*, 2018.

Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. STD: Sparse-to-dense 3D object detector for point cloud. In *ICCV*, 2019.

Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3DSSD: Point-based 3D single stage object detector. 2020.

Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3D object detection and tracking. *CoRR*, abs/2006.11275, 2020.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. CutMix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, 2019.

Xiaosong Zhang, Fang Wan, Chang Liu, Rongrong Ji, and Qixiang Ye. FreeAnchor: Learning to match anchors for visual object detection. In *NeurIPS*, 2019.

Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017.

Pan Zhou, Jiashi Feng, Chao Ma, Caiming Xiong, Steven Chu-Hong Hoi, and Weinan E. Towards theoretically understanding why sgd generalizes better than adam in deep learning. In *NeurIPS*, 2020.

Yin Zhou and Oncel Tuzel. VoxelNet: End-to-end learning for point cloud based 3D object detection. In *CVPR*, 2018.

Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-balanced grouping and sampling for point cloud 3D object detection. *CoRR*, abs/1908.09492, 2019. URL `http://arxiv.org/abs/1908.09492`.

Barret Zoph, Ekin D. Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens, and Quoc V. Le. Learning data augmentation strategies for object detection. *CoRR*, abs/1906.11172, 2019.

A RELATED WORK

3D object detection. Many approaches (Yang et al., 2019; Shi et al., 2019; Yan et al., 2018; Liu et al., 2020) have been focusing on processing LiDAR point cloud to improve the performance of 3D object detection. To deal with the irregular and unstructured nature of point cloud, common approaches either apply convolutional neural network (CNN) to the voxelized representation (Zhou & Tuzel, 2018; Yan et al., 2018; Lang et al., 2019; Liu et al., 2020; Deng et al., 2020) or process raw points (Shi et al., 2019; Qi et al., 2019a; Yang et al., 2020) by PointNets (Qi et al., 2017a;b). Recent methods (Shi et al., 2020; Yang et al., 2019; Chen et al., 2019c) exploit both voxel representation and raw points. There are also attempts (Wang et al., 2019; Chen et al., 2018) that purely rely on cameras for 3D detection.

Previous works aggregate image and point cloud features from different views (Chen et al., 2017; Ku et al., 2018); the efficiency is limited by the view aggregation for a large quantity of anchors (Ku et al., 2018) or the proposals (Chen et al., 2017). More recent works (Liang et al., 2019; 2018; Sindagi et al., 2019) fuse image features into each point, but they exhibit various feature misalignment issues. For example, MVX-Net (Sindagi et al., 2019) quantizes image coordinates, while methods (Liang et al., 2019; 2018) based on ContFuse (Liang et al., 2018) use the nearest points for each BEV feature grid. Frustum-based methods (Qi et al., 2018; Wang & Jia, 2019) obtain frustum proposals from an image and then apply PointNet (Qi et al., 2017a) to point cloud for 3D object localization. Their performance is limited by the proposal qualities and they may not fully exploit the complementary information of multi-modalities. ImVoteNet (Qi et al., 2020) skips the above-mentioned issue by fusing 2D votes in images and 3D votes in point clouds.

Augmentations for detection. Data augmentation is crucial to improve the models' performance. However, the current single-modality 3D detectors (Yang et al., 2019; Shi et al., 2019; 2020) use more aggressive data augmentations than existing multi-modality methods (Qi et al., 2018; Liang et al., 2019; Sindagi et al., 2019) do. Conventional image augmentations include but are not limited to random cropping, random flipping, and multi-scale training (He et al., 2016; Chen et al., 2019b). For point cloud data, common augmentation techniques are random flipping, rotation, translation, and scaling (Yan et al., 2018; Zhou & Tuzel, 2018; Shi et al., 2019). In multi-modality 3D detection, the practices of augmentations vary across different methods (Liang et al., 2018; 2019; Sindagi et al., 2019). For example, ContFuse (Liang et al., 2018) skips random flip, whereas MVX-Net (Sindagi et al., 2019) does not apply random rotation and translation to maintain consistency between image and point cloud.

There are augmentations applied to regions that contain objects of an image (Devries & Taylor, 2017; Yun et al., 2019; Fang et al., 2019). A representative method is to cut and paste objects (Dwibedi et al., 2017; Dvornik et al., 2018; Fang et al., 2019). While Dwibedi *et al*. (Dwibedi et al., 2017) paste objects randomly, recent works use a location probability map (Fang et al., 2019), semantic and depth information (Georgakis et al., 2017), or a visual context model (Dvornik et al., 2018) to guide the pasting process. Cutting and pasting the points of objects is also common for LiDAR-based 3D detection methods (Yan et al., 2018; Liu et al., 2020; Yang et al., 2019) but is absent from multi-modality methods (Qi et al., 2018; Liang et al., 2019; Sindagi et al., 2019).

B DISCUSSION

Applications to different modalities. Multi-modality transformation flow and MoCa apply to multi-modality methods that take LiDAR point cloud, range view or depth image, and RGB image as their inputs. For depth or range view images, as long as the data augmentation can be applied to depth or range view images and the data augmentation has reversible transformation, transformation flow can be used with these data augmentations to maintain consistency in multi-modality fusion. For example, as rotation, flip, and cut and paste can be applied to range view images (Fan et al., 2021), transformation flow can also work to maintain the correspondence to fuse the features from range view images and RGB images. When range view or depth images are used, MoCa can also cut and paste the patches of objects from range view or depth images. We can still handle the occlusion in 3D space by first projecting the depth to 3D space from these images.

Limitations and improvements. For simplicity, MoCa pastes the objects to the same positions in a new scene as to where they are in their original scenes. The flexibility of MoCa can be improved

by making the pasting process more editable, e.g., enabling to decide the position of the objects in a new scene. A more editable pasting process could increase the diversity of the augmented scenes and may improve the model's generalizability because it prevents the model from overfitting to objects at certain locations. Furthermore, we notice that the pasted scenes are not as real as natural scenes because we only blend the image patches when pasting them to a new image. Therefore, graphic methods like rendering engines may further improve the performance by improving the realness of the augmented training samples.

C Implementation Details

Due to the complexity of multi-modality 3D object detection, we discuss the critical details in our implementation to help avoiding pitfalls in coordinate transformation, data augmentation, and model training.

Projection from LiDAR to image. For feature fusion, the point cloud is first transformed from LiDAR coordinates P_{lidar} to camera coordinates and then projected to image pixel coordinates P_{img}. On KITTI dataset (Geiger et al., 2012), the projection is calculated as follows:

$$P_{img} = P^0_{rect} R^0_{rect} T_{cam \leftarrow lidar} P_{lidar}, \tag{3}$$

where $T_{cam \leftarrow lidar}$ is the transformation matrix from LiDAR coordinates to camera coordinates, R^0_{rect} is the rectifying rotation matrix of the left camera, and P^0_{rect} is the calibration matrix of the left camera.

On nuScenes dataset (Caesar et al., 2019), the points in LiDAR coordinates are first transformed to the ego car's global coordinates since the LiDAR (20Hz) and camera (12Hz) work at different frequencies. Therefore, the transformation is calculated as follows:

$$P_{img} = T_{cam \leftarrow ego} T_{ego_c \leftarrow ego_l} T_{ego \leftarrow lidar} P_{lidar}, \tag{4}$$

where $T_{ego \leftarrow lidar}$ is the transformation matrix from LiDAR to the ego pose at timestamp t_l when the LiDAR frame is recorded, $T_{ego_c \leftarrow ego_l}$ is the transformation matrix from ego pose at timestamp t_l to the ego pose at timestamp t_c when the camera frame is captured, and $T_{cam \leftarrow ego}$ is the transformation matrix from ego pose at timestamp t_c to the camera.

Multi-modality transformation flow. When training single-modality detectors, the model is trained on the augmented data with the augmented label, i.e., if the input data is flipped, its training label should also be flipped. The single-modality detector should obey this rule even if they are trained inside a multi-modality detector (we should not make the network learn to predict a box at the left side if the object is in the right side of the image). During fusion, transformation flow allows the points in a modality to correctly find its corresponding point in another modality. For example, given a flipped point cloud data and its unflipped image, the image detector should be trained with unflipped 2D labels and the point cloud detector should be trained with flipped 3D labels. During fusion, transformation flow allows a point of an object in the right side of LiDAR coordinates to find its true correspondence, which should be a pixel of the object in the left side of the image. This could improve the robustness of feature fusion because it forces the detector to exploit the semantic feature that should be invariant to transformations. On the nuScenes dataset, using un-synchronized flipping of the two modalities slightly improves the performance (0.5%NDS).

Multi-modality cut and paste. Multi-modality cut and paste (MoCa) needs ground-truth instance masks to paste image patches of objects, as in (Dwibedi et al., 2017; Dvornik et al., 2018; Fang et al., 2019). For KITTI dataset, we use the instance masks in KINS dataset (Qi et al., 2019b) to build the multi-modality GT database. For nuScenes dataset, we use pseudo instance masks predicted by an instance segmentation model. Specifically, we first train an HTC (Chen et al., 2019a) with ResNeXt 32×4d backbone on COCO dataset with 3× schedule (Chen et al., 2019b). Then we fine-tune the HTC on nuImages dataset by 20 epochs and use that model to predict instance masks of images in nuScenes dataset. Note that we *only use the objects in the training split to formulate the GT database in all experiments*. No information from validation/test split is used in training.

Training details. On KITTI dataset, both SECOND (Yan et al., 2018) and MVX-Net (Sindagi et al., 2019) are trained by 80 epochs with a batch size of 16. We adopt a half-period cosine schedule (Loshchilov & Hutter, 2017) for learning rate decaying and use a linear warm-up strategy in the first 1K iterations. The initial learning rate is 0.003 for all the 3D detectors that use ADAM (Kingma

Table 5: Ablation study of multi-modality copy-paste

(a) Comparison of different IoF thresholds on Multi-modality 3D detector. 'No test' means not applying collision test, and 'mixed' means using different thresholds during training

Threshold	3D mAP (%)		
	Easy	Mod.	Hard
No test	79.9	69.0	64.5
0.3	80.9	69.7	66.0
0.5	80.0	69.6	**66.0**
0.7	80.0	69.7	65.6
Mixed	**80.9**	**70.2**	65.2

(b) Comparison of different IoF thresholds on 2D detector. 'No test' means not applying collision test, and 'mixed' means using different thresholds during training

Threshold	2D mAP (%)		
	Easy	Mod.	Hard
No test	84.9	78.2	72.1
0.3	86.1	79.0	73.9
0.5	85.8	79.0	73.6
0.7	86.8	78.7	73.2
Mixed	86.3	**79.3**	**74.4**

(c) Comparison of different augmentations and different epochs on Faster R-CNN. 'Baseline'– not using cut and paste. and 'AutoAug'– searched augmentation strategies

Method	Epochs	2D mAP (%)		
		Easy	Mod.	Hard
Baseline	20	86.3	75.2	71.8
	36	86.6	73.3	67.4
AutoAug	20	86.4	78.5	73.7
	36	84.9	78.2	72.1
MoCa (ours)	20	86.3	79.3	**74.4**
	36	**86.6**	**79.5**	74.1

& Ba, 2015) optimizer. When retaining the original optimizers (Sec. 2.3) for MVX-Net, we train the image branch using SGD optimizer with momentum and the initial learning rate is 0.05. The hyperparameters for point cloud branch remain the same as those for SECOND.

For nuScenes dataset (Caesar et al., 2019), both PointPillars (Lang et al., 2019) and MVX-Net (Sindagi et al., 2019) are trained by 20 epochs with batch sizes of 32 and 16, respectively. Notably, this is different from the official implementation (Caesar et al., 2019), where PointPillars (Lang et al., 2019) is trained by 125 epochs, which costs much time. We adopt step learning rate decaying schedule following the practice in mmdetection (Chen et al., 2019b), *i.e.*, the learning rate is decayed by 0.1 after the 16th and 19th epoch, respectively. The initial learning rate is 0.001 for all the 3D detectors using ADAM (Kingma & Ba, 2015) optimizer. When training MVX-Net, we also retains the original optimizers, where the image branch is trained by SGD optimizer with momentum and the initial learning rate is 0.0012. The hyperparameters for point cloud branch remain the same as those for PointPillars.

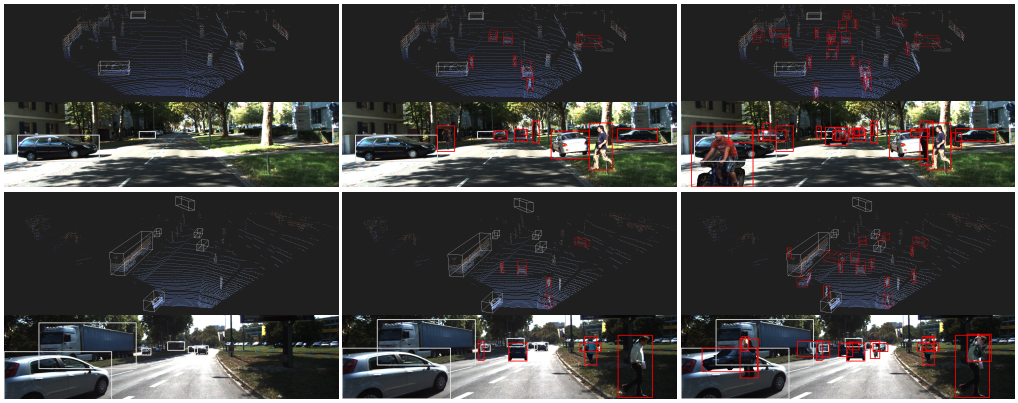
Since one training sample for MVX-Net contains six images from multiple views and one LiDAR point cloud frame, each GPU can only contain one training sample during each iteration. This degrades the performance siginificantly because the batch size in one GPU is so small that the inaccurate statistics in Batch Normalization (BN) (Ioffe & Szegedy, 2015) affects the training process. Therefore, we use Syncronized Batch Normalization (SyncBN) (Liu et al., 2018) to solve this issue. We report all the results of our methods using SyncBN on nuScenes dataset. We do not use SyncBN for models on KITTI dataset because the batch size is 2 in each GPU as there is only one image and one LiDAR point cloud frame in one training sample.

D More Experiments

Multi-modality cut and paste (MoCa). We evaluate the effectiveness of components in MoCa. We empirically find that having 6 pedestrians, 6 cyclists, and 12 cars in a frame for training yields the best performance. Table 5a and Table 5b show that collision test improves the detector's performance of both multi-modality 3D detector and 2D detector, and different thresholds lead to different APs under different conditions. The proposed mixed IoF thresholds yield the best performance and is adopted in other experiments.

We further compare MoCa with other data augmentation techniques for 2D detectors. We train Faster R-CNN (Ren et al., 2015) with FPN (Lin et al., 2017) following the standard setting (Chen et al., 2019b) except that we use 20 or 36 epochs and different augmentation strategies. Cut and paste significantly improves the results, in comparison with those not using cut and paste and those using searched augmentation strategies (Zoph et al., 2019) (Table 5c). As the training schedule becomes longer, cut and paste maintains its high performance, while the performances of baseline and AutoAug (Zoph et al., 2019) degrade. The results of sustaining a longer training suggest the effectiveness of MoCa in reducing overfitting.

We provide more visual results in Figure 6. The results show that MoCa provides visually reasonable results after collision test. The new frames are enriched by more objects from less-frequent categories such as pedestrian.



Original frame | Final frame | Frame w/o collision test

Figure 6: **Results of MoCa.** The figure is best seen in color with zoomed in.

Table 6: Comparison of different training strategies. 'Pre-train' indicates which pre-trained model is used. 'COCO' means the model is pre-trained on COCO dataset, and '+ KITTI' means the model is further pre-trained on KITTI 2D dataset. 'Freeze' indicates whether ResNet-50 (He et al., 2016) in the image branch is frozen. The average results and standard deviation over 5 runs are reported. The best results in each setting are bolded

Pre-train	Freeze	Optimizer	3D mAP (%)		
			Easy	Mod.	Hard
COCO	×	SGD	80.2 ± 1.0	69.9 ± 0.8	66.1 ± 1.1
		ADAM	80.9 ± 0.7	70.2 ± 0.7	66.3 ± 1.2
		Hybrid	$\mathbf{81.1 \pm 0.7}$	$\mathbf{71.0 \pm 0.6}$	$\mathbf{67.2 \pm 0.6}$
COCO + KITTI	×	SGD	80.5 ± 0.8	70.3 ± 0.2	67.0 ± 0.4
		ADAM	80.6 ± 0.3	69.6 ± 0.3	65.7 ± 0.6
		Hybrid	$\mathbf{81.1 \pm 0.5}$	$\mathbf{70.6 \pm 0.7}$	$\mathbf{67.0 \pm 1.0}$
COCO + KITTI	✓	SGD	80.3 ± 0.6	70.3 ± 0.4	66.5 ± 0.8
		ADAM	80.9 ± 0.6	70.7 ± 0.5	67.0 ± 0.3
		Hybrid	$\mathbf{81.8 \pm 0.4}$	$\mathbf{71.2 \pm 0.5}$	$\mathbf{67.8 \pm 0.7}$

Retaining the original optimizers. As discussed in Sec. 2.3, the choice of optimizer for the image feature extractor matters. To verify the versatility of retaining the original optimizers and find a good training strategy, we adopt three training strategies for pre-training image feature extractors and jointly training multi-modality detectors.

Here, we compare the following variants: (1) *SGD*, the SGD optimizer is used for the image-branch pre-training as well as for both the image and point cloud branches during joint training; (2) *ADAM*, the SGD optimizer is used for the image-branch pre-training but the ADAM optimizer is used for both the image and point cloud branches during joint training; (3) *Hybrid*, we retain the original optimizer of each modality branch, as presented in Sec. 2.3. The first strategy trains a Faster R-CNN (Ren et al., 2015) on COCO2017 dataset by the multi-scale 3× schedule (Chen et al., 2019b; Wu et al., 2019) with ResNet-50 (He et al., 2016) pre-trained on ImageNet (Russakovsky et al., 2015). Then we train Faster R-CNN on the subset of COCO training split that only contains three classes: people (for pedestrian in KITTI), bicycle (for cyclist in KITTI), and car, following Frustum-PointNet (Qi et al., 2018). The weights of ResNet-50 and FPN pre-trained in the Faster R-CNN are adopted in the image branch of the multi-modality detector for joint training. The second strategy further fine-tunes the Faster R-CNN using *MoCa* for 20 epochs with mixed IoF thresholds (last row in Table 5a) before joint training. The third strategy uses a similar pre-training strategy as the SECOND one but freezes the ResNet-50 backbone in joint training. The optimal learning rate and other hyper-parameters for all the aforementioned variants are found using a grid search to ensure fair comparisons.

As shown in Table 6, retaining the original optimizers is beneficial, and the synergy of the third strategy and retaining the original optimizers works best among these training and optimization strategies. Our results here are purely empirical but they reveal the interesting tendency of using different optimizers in different modality branches. This phenomenon worths further exploration especially from the perspective of optimization routes (Keskar & Socher, 2017; Zhou et al., 2020).

Table 7: Comparison of image branches pre-trained by different detectors

Detector	Faster R-CNN	Mask R-CNN	Cascade	HTC
NDS (%)	57.4	57.6	57.9	**58.1**

Table 8: Ablation study of each component with PointPainting (Vora et al., 2020) on nuScenes validation set. Modifications are added sequentially

Method	NDS (%)	mAP (%)
PointPillars (Lang et al., 2019) + FPN (Lin et al., 2017)	53.4	40.1
+ image branch (Zhao et al., 2017)	56.5	45.1
+ MoCa	56.8	45.4
+ retain original optimizers	57.2	45.8

Pre-training on nuScenes dataset. We observe different effectiveness of the image branch in our experiments when it is pre-trained in Faster R-CNN (Ren et al., 2015), Mask R-CNN (He et al., 2017), Cascade Mask R-CNN (Cai & Vasconcelos, 2018), and HTC (Chen et al., 2019a) (Table 7). Using similar backbones and necks, the image branch from HTC pre-trained on nuImages dataset (Caesar et al., 2019) shows a gain of 0.7% NDS against that from Faster R-CNN. Thus, we adopt the image branch from HTC in other experiments on nuScenes dataset.

The image branch from HTC pre-trained on nuImages dataset shows a gain of 0.7% NDS against that from Faster R-CNN as shown in Table 7. Those pre-trained detectors are first trained by 20 epochs using the standard setting on COCO dataset and released by MMDetection (Chen et al., 2019b). We further fine-tune those detectors on the nuImages dataset by 20 epochs using similar hyper parameters as those for the COCO dataset. When jointly training the multi-modality detectors, the weights in ResNet-50 backbone and FPN of those pre-trained detectors are adopted to initialize the image branch.

Step-by-step results with PointPainting. We show that our method generalizes to more recent PointPainting (Vora et al., 2020). PointPainting fuses image segmentation predictions with points for multi-modality 3D object detection. We use PSPNet (Zhao et al., 2017) with ResNet-18 backbone as the image branch, which is pre-trained on nuImages dataset by 80000 iterations using the standard settings in MMSegmentation (Contributors, 2020). As the fusion technique in Section 3.3 relies on FPN-based object detectors, we do not study it here. As shown in Table 8, adding PSPNet (Zhao et al., 2017) as the image segmentation branch brings improvement of 2.1% NDS. MoCa further improves the performance by 0.3% NDS, and retaining original optimizer gains 0.4% NDS. The results verifies the generalizability of MoCa and retaining the original optimizer.

It is worth noting that the official code of PointPainting is not available publicly. Thus, the improvement by our method may also be affected by missing some important implementation details. We re-implemented the method with our best effort. The PointPainting authors use a higher baseline (55% NDS on test set, 54.5% NDS on val set) and the training recipe of the segmentation network and multi-modality detector is unknown. Though PSPNet should have stronger representation ability and performance than FCN, the improvement brought by PSPNet trained by us (+5 AP) is smaller than that brought by FCN in the original PointPainting (+6.3 AP). The hyper parameters might need delicate tuning as we adopt the same hyper-parameters of MoCa and optimization as those for MVX-Net due to limited resources.

Stronger PointPillars baselines. We adopt stronger PointPillars baselines in MVX-Net to verify our method's generalizability by enhancing the head, backbone, and training schedule of PointPillars. For completeness, we also put the detailed step-by-step results in Table 9. Notably, our PointPillars baseline already achieves very high performance on the validation set. However, our methods consistently improve performance, especially on challenging classes for LiDAR-based detectors, such as bicycles, motorcycles, traffic cones, and pedestrians.

Table 9: Comparison with previous methods and stronger PointPillars baselines on nuScenes validation set. 'Con. Veh.', 'Ped.', and 'T.C.' are the abbreviations of construction vehicle, pedestrian, and traffic cone, respectively. 'FA' means FreeAnchor and '3×' means longer training schedule. NDS score, mAP, and APs of each categories are reported. The single class AP not reported in the paper is marked by '-'. The best results are bolded

Method	Modality	NDS	mAP	Car	Truck	Bus	Trailer	Con. Veh.	Ped.	Motor	Bicycle	T.C.	Barrier
PointPillars	L	46.8	28.2	75.5	31.6	44.9	23.7	4.0	49.6	14.6	0.4	8.0	30.0
3D-CVF	L + I	49.8	42.2	79.7	37.9	55.0	36.3	-	71.3	37.2	-	40.8	47.1
PointPillars+FPN	L	53.4	40.1	80.6	35.9	43.5	29.2	5.4	71.9	34.9	11.8	35	52.6
3DSSD	L	56.4	42.6	81.2	47.2	61.4	30.5	12.6	70.2	36.0	8.6	31.1	47.9
PointPainting	L + I	58.1	46.4	77.9	35.8	36.1	37.3	15.8	73.3	41.5	24.1	62.4	60.2
CenterPoint	L	65.0	56.6	84.6	54.7	66	32.3	15.1	84.5	56.9	38.6	67.4	66.1
MoCa + MVX-Net++	L+I	58.1	47.9	82.4	41.5	49.6	28.6	9.1	79.1	50.3	27.2	49	61.9
Stronger baselines based on PointPillars+FPN													
+ FA		55.1	43.7	81.5	40.0	50.0	29.4	9.2	74.3	44.5	16.5	39.6	52.4
+ FA + RegNetX-400MF		56.7	45.5	82.0	41.9	50.7	32.3	11.0	75.4	50.1	19.1	44.1	48.8
+ FA + RegNetX-1.6GF	L	61.2	51.4	83.2	48.2	60.5	30.4	16.6	78.1	59.4	25.9	49.1	62.3
+ FA + RegNetX-3.2GF		62.2	52.1	83.6	51.1	62.3	36.0	17.3	78.2	56.1	24.7	50.0	62.0
+ FA + RegNetX-3.2GF + 3×		64.2	56.9	85.5	54.9	66.8	35.4	22.2	81.2	62.4	35.6	59.2	65.4
MoCa + MVX-Net++ on stronger PointPillars baselines													
+ FA		60.3	52.9	83.6	48.5	56.4	31.4	10.8	81.6	61.0	35.7	58.1	61.7
+ FA + RegNetX-400MF		62.1	55.2	84.2	51.7	63.6	34.2	18.0	82.0	61.8	32.9	59.4	64.0
+ FA + RegNetX-1.6GF	L + I	64.7	58.4	85.2	55.5	64.5	35.0	20.8	84.5	68.0	42.8	62.3	65.1
+ FA + RegNetX-3.2GF		65.4	59.2	85.7	56.8	66.2	35.8	21.7	84.4	67.4	44.2	62.8	66.3
+ FA + RegNetX-3.2GF + 3×		**67.8**	**62.5**	**87.0**	**60.8**	**68.2**	**39.7**	**23.9**	**85.8**	**70.2**	**51.6**	**68.1**	**69.5**