Cross-Domain Transfer Learning for 3D Object Detection

Dechao Zhang(12011220) Ziyang Leng(12011513)

1 Overview

Deep learning models trained on annotated datasets (source domain) are often observed with considerable accuracy decrease on other datasets (target domain). However, retraining the model on the target domain is of great cost. To achieve efficient domain adaptation, we use self training to modify the pesudo labels generated by the source domain. But there exists noise during self training, which makes training less effective. Our research objective is to reduce the noise caused by self training.

3D object detection targets the action of taking input from sensor (*e.g.* LiDAR point clouds) then categorize and localize objects. This is one of the major two detection solutions for autonomous driving. Recently with the success of deep neural networks benefiting various fields such as robotics, natural language processing and computer vision, etc. Autonomous cars trained and equipped with these networks that perform the 3D object detection can perform the perception with particularly low latency and high accuracy.

However, deep learning models trained on one annotated dataset(source domain) are often observed with considerable accuracy degeradation on other dataset(target domain), especially with two datasets collected in different countries [1]. Previous research has shown that the primary adaption hurdle to overcome are differences in object size, especailly cars across geographic domains [1]. It seems unpratical for automobile manufacturers to restrict their autonomous driving product only operates on certain areas. And due to the enormous costs for collection and annotation of datasets it's necessary to come up with solutions that can adapting the 3D detector trained on labeled source domain to unlabeled target domain, which is known as unsupervised domain adaption(UDA) for 3D detection.

Recent researches on unsupervised domain adaption comes up with several solutions which all have their benifits but shortcomings. Methods which use existing statistic data of object size in different domains to eliminate the size bias from source domain, either adjusting the input size from source domain or correcting the output of target domain, requires large mount of data which may be infeasible or inaccurate in many areas [1]. It's the same situation with finetuning the pre-trained model with smaller amount of labeled data from target domain. Other solutions that rely on popular and more mature 2D unsupervised domain adaption methods which leverage feature alighment techniques [2] [3] [4] to mitigate domain gap become ineffective with 3D point clouds without the feature information such as color, lighting and texture. Giving the restrictions on above solutions, self-training recently shown itself as an effective approach to UDA.

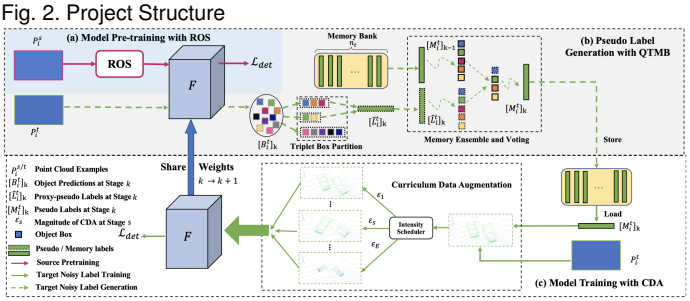
Fig. 1. Testing data from [1]

Setting	Source\Target	KITTI	Argoverse	nuScenes	Lyft	Waymo
Easy	KITTI	**88.0 / 82.5**	55.8 / 27.7	47.4 / 13.3	81.7 / 51.8	45.2 / 11.9
	Argoverse	69.5 / 33.9	**79.2 / 57.8**	52.5 / 21.8	86.9 / 67.4	83.8 / 40.2
	nuScenes	49.7 / 13.4	73.2 / 21.8	**73.4 / 38.1**	89.0 / 38.2	78.8 / 36.7
	Lyft	74.3 / 39.4	77.1 / 45.8	63.5 / 23.9	**90.2 / 87.3**	87.0 / 64.7
	Waymo	51.9 / 13.1	76.4 / 42.6	55.5 / 21.6	87.9 / 74.5	**90.1 / 85.3**
Moderate	KITTI	**80.6 / 68.9**	44.9 / 22.3	26.2 / 8.3	61.8 / 33.7	43.9 / 12.3
	Argoverse	56.6 / 31.4	**69.9 / 44.2**	27.6 / 11.8	66.6 / 42.1	72.3 / 35.1
	nuScenes	39.8 / 10.7	56.6 / 17.1	**40.7 / 21.2**	71.4 / 25.0	68.2 / 30.8
	Lyft	61.1 / 34.3	62.5 / 35.3	33.6 / 12.3	**83.7 / 65.5**	77.6 / 53.2
	Waymo	45.8 / 13.2	64.4 / 29.8	28.9 / 13.7	74.2 / 53.8	**85.9 / 67.9**
Hard	KITTI	**81.9 / 66.7**	42.5 / 22.2	24.9 / 8.8	57.4 / 34.2	41.5 / 12.6
	Argoverse	58.5 / 33.3	**69.9 / 42.8**	26.8 / 14.5	64.4 / 42.7	68.5 / 36.8
	nuScenes	39.6 / 10.1	53.3 / 16.7	**40.2 / 20.5**	67.7 / 25.7	66.9 / 29.0
	Lyft	60.7 / 33.9	62.9 / 35.9	30.6 / 11.7	**79.3 / 65.5**	77.0 / 53.9
	Waymo	46.3 / 12.6	61.6 / 29.0	28.4 / 14.1	74.1 / 54.5	**80.4 / 67.7**
0-30m	KITTI	**88.8 / 84.9**	58.4 / 34.7	47.9 / 14.9	77.8 / 54.2	48.0 / 14.0
	Argoverse	74.2 / 46.8	**83.3 / 63.3**	55.3 / 26.9	87.7 / 69.5	85.7 / 44.4
	nuScenes	50.7 / 13.9	73.7 / 26.0	**73.2 / 42.8**	89.1 / 43.8	79.8 / 43.4
	Lyft	75.1 / 45.2	81.0 / 54.0	61.6 / 25.4	**90.4 / 88.5**	88.6 / 70.9
	Waymo	56.8 / 15.0	80.6 / 48.1	57.8 / 24.0	88.4 / 76.2	**90.4 / 87.2**
30m-50m	KITTI	**70.2 / 51.4**	46.5 / 19.0	9.8 / 4.5	60.1 / 34.5	50.5 / 21.4
	Argoverse	33.9 / 11.8	**72.2 / 39.5**	9.5 / 9.1	65.9 / 39.1	75.9 / 42.1
	nuScenes	24.1 / 3.8	46.3 / 6.4	**17.1 / 4.1**	70.1 / 18.9	69.4 / 29.2
	Lyft	39.3 / 16.6	59.2 / 21.8	11.2 / 9.1	**83.8 / 62.7**	79.4 / 55.5
	Waymo	31.7 / 9.3	58.0 / 18.8	9.9 / 9.1	74.5 / 51.4	**87.5 / 68.8**
50m-70m	KITTI	**28.8 / 12.0**	9.2 / 3.0	1.1 / 0.0	33.2 / 9.6	27.1 / 12.0
	Argoverse	10.9 / 1.3	**29.9 / 6.9**	0.5 / 0.0	35.1 / 14.5	46.2 / 23.0
	nuScenes	6.5 / 1.5	15.2 / 2.3	**9.1 / 9.1**	41.8 / 5.3	37.9 / 15.2
	Lyft	13.6 / 4.6	23.1 / 3.9	1.1 / 0.0	**62.7 / 33.1**	54.6 / 27.5
	Waymo	5.6 / 1.8	26.9 / 5.6	0.9 / 0.0	50.8 / 21.3	**63.5 / 41.1**

2 Model

To realize self training, we use the ST3D [5] model.

Fig. 2. Project Structure



2.1 Domain Randomization Pre-trainig

Giving the fact that 3D bounding boxes have a total 7 degrees of freedom, namely the object size (l, w, h), the object center location (c_x, c_y, c_z) and orientation angel θ,

we can perform source domain randomization by rendering random scale factors on object size(r_l, r_w, r_h). Therefore considering points in the orginal source domain, where position for point p_i can be expressed as $(p_i^l, p_i^w, p_i^h) = (p_i^x - c_x, p_i^y - c_y, p_i^z - c_z) \cdot R$, where R is tranformation matrix on to the local coordinate system. By producing dot product using random scale factor on points, the point coordinate in the ground reference system can be expressed as

$$p_i^r = (r_l p_i^l, r_w p_i^w, r_h p_i^h) \cdot R^T + (c_x, c_y, c_z)$$

2.2 Pseudo Label Generated from QTMB

QTMB consists of 4 submodels, and takes input from teacher model in the mean-teacher pardigm. An assessing criterion for prediction boxes based on IoU regression that uses two fully connected layer to predict IoU between RoIs and ground truth labels. A triplet box partition used to avoid ambiguous prediction. That is setting a standard which partition the above assessing scores into basically three section, the positive, negative and neutral ones by comparing its value with $T_(pos)$ and $T_(neg)$. While the positive prediction can help generating the pseudo labels, the negative prediction avoid the detector from generating high uncertainty pseudo lables, the neutral ones are disgarded for their possible abiguousity caused to the label generating system. Also since this is a memory bank, the generated pseudo lables are not directly feed into the memory, with matching the object bounding boxes with similar attributes from both the memory and newly generated label set and merge them into new boxes, this can greatly reduce the possibilities of leading the final bounding box into unreasonable conditions. For those boxes that haven't been matched during the third process, another asssessing criterion based on the memory bank decides their fates. With the history of unmatched boxes stored in memory bank, we can pick up the unmatched boxes with successive history, which indicates that they may be unexpectedly missed but still may be valueble, and again adding them to the memory bank.

2.3 Curriculum Data Augmentation

We decided to divide the training epochs into E stages, with the first stage being with the most simple tasks, stages later are intensified with data argmentation according to a ratio of the further training completion.

3 Code Skeleton

Our code is based on OpenPCDet v0.2 [6]. We mainly use the pecdet package, modify the package according to our model and add a tools package. In this stage, we don't incorporate our model with mean teacher paradigm.

3.1 Random Object Scaling(ROS)

In pcdet/datasets/augmentor/data-augmentor.py, we mainly realize it in the function "random-object-scaling". The function calls function "scale-pre-object" in augmentor-utils. "scale-pre-object" is designed to uniformly sacle object with given range. The "gt-boxes" contains the information of the annotated 3D bounding box. Information contains center, size and heading angle in sequence. The "points" contains the information of points in lidar. The "gt-boxes-mask" contains the information of boolen mask for gt-boxes. The "scale-perturb" is about the noise of the bounding box's size.

For (r_l, r_w, r_h), we just use the scale-perturb as a seed to call the np.random and then generate the random scale factors. To generate the new size, we just multiply the original size with (r_l, r_w, r_h) using matrix multiplication. And then follow the matrix operation as mentioned above, we can get the newly generated size, center as we want.

3.2 Quality-aware Triplet Memory Bank(QTMB)

3.2.1 IoU-based Quality-aware Criterion for Scoring

Its realization is mainly in pcdet/utils/self-training-utils.py. The function "save-pseudo-label-batch" is designed to save pseudo label for give batch. If model is given, use model to inference pred-dicts, otherwise, directly use given pred-dicts. For its argument,

input-dict: batch data read from dataloader.

pred-dicts: Dict if not given model. predict results to be generated pseudo label and saved.

need-update: Bool. If set to true, use consistency matching to update pseudo label.

IoU prediction score u_b is given by the "pred-dicts" and stored in the "pred-scores". And after comparing it with $T_(pos)$ and $T_(neg)$. We can label the pseudo box with positice, negative or ignored.

To further look at how the "pred-dicts" is generated, we find that "save-pseudo-label-epoch" calls the "save-pseudo-label-batch". In "save-pseudo-label-epoch", "pred-dicts" is given by model(target-batch). "Model" is given by its caller. The "model" definition is in the tools/train.py.

3.2.2 Memory Update and Pseudo Label Generation

Its realization is mainly in pcdet/utils/memory-ensemble-utils.py. The function "consistency-ensemble" is written to do the memory ensembling and memory voting operation. For its argument,

gt-infos-a

gt-boxes: (N, 9) [x, y, z, dx, dy, dz, heading, label, scores] in LiDAR for previous pseudo boxes

cls-scores: (N)

iou-scores: (N)

memory-counter: (N)

gt-infos-b

gt-boxes: (M, 9) [x, y, z, dx, dy, dz, heading, label, scores] in LiDAR for current pseudo boxes

cls-scores: (M)

iou-scores: (M)

memory-counter: (M)

memory-ensemble-cfg: config setting

Returns:

gt-infos

gt-boxes: (K, 9) [x, y, z, dx, dy, dz, heading, label, scores] in LiDAR for merged pseudo boxes

cls-scores: (K)

iou-scores: (K)

memory-counter: (K)

To get pair-wise 3D IoU matrix A, we use iou3d-nms-utils.boxes-iou3d-gpu. It's a function in pcdet/ops/iou3d-nms/iou3d-nms-utils given by OpenPCDet v0.2. If the IoU is bigger than IOU-THRESH, we consider them as matching boxes. And after matching boxes by IoU, use box in the pair

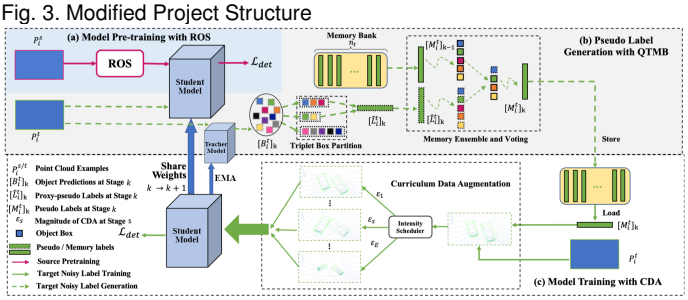
with higher confidence, which is u_b calculated in "Memory Update and Pseudo Label Generation". For matching pairs, we need to clear the ignore counter. For pairs whose IoU is less than 0.1, we do the memory voting operation. We update the ignore-count for unmatched boxes in the memory bank by adding 1 and initialize the UMC of the newly generated boxes as 0. We ignore gt-boxes that ignore-count == IGNORE-THRESH, remove gt-boxes that ignore-count $\geq$ RM-THRESH and add newly appeared boxes.

3.3 Curriculum Data Augmentation(CDA)

Its realization is mainly data-augmentor in pcdet/datasets/augmentor. The function "re-prepare" contains the realization of CDA. To scale data augmentation intensity, it calls the function "adjust-augment-intensity". In the function "adjust-augment-intensity", it has a internal function "cal-new-intensity" and use a num–flag to distinguish between "random-world-scaling" and "random-world-rotation" (1 for "random-world-scaling" and 0 for "random-world-rotation").

To reduce noise caused by self training, we combine ST3D model with mean teacher paradigm.

Fig. 3. Modified Project Structure



3.4 Mean Teacher

The mean teacher paradigm [7] consists of two models, a student model F and a teacher model F′, both equipped with identical network structure but different weight θ and θ'. And the teacher model weight is updated from student model using exponential moving average,

$$\theta' = m\theta' + (1-m)\theta$$

m here is momentum often set close to 1. Therefore with the supervision of teacher model on student model by supplying high quality pseudo labels, and student model gradually leads teacher model into certain state, the two models will finally merge to consistency.

3.5 Code Modification

First, we make a copy of our original model to initialize the teacher model. Second, we use the student model parameters and the last time teacher model parameters to update the teacher model. In the beginning, the teacher model doesn't have an ideal effect. Therefore, the momentum m should be relatively small at start and become large when epoch number increases. Last, we use the teacher model to generate pesudo label instead of the student model.

4 Experiments

4.1 Experiments Setup

Datasets. According to our project plan, we conduct unsupervised domain adaptation from nuScense [8] to KITTI [9] , mainly focus on the cross domain with different number of LiDAR beams. This time we add the mean teacher paradigm to evaluate if our idea of implementing it will work and improve overall precision of final results. Also another part of our plan is to test out the UDA from nuScense to Suscape, which is our own dataset.

Comparison. By evaluate the average precision of *Source Only* model from nuScenes dataset on KITTI and each check point during self training process before and after adding mean teacher. Also we bring more comparison between the teacher model and student model through the whole training process.

Evaluation Metric. We compared AP of car category with R11 and R40 which use average of 11 and 40 equidistant recall precisions AP, and evaluate the label of three difficulties, namely *easy*, *moderate* and *hard* under the IoU threshold of corresponding 0.7, 0.5 and 0.5, which is consistent with the KITTI evaluation metric. And perform evalutation on four types of detection:

1) bbox: 2D detection precision
2) bev: Bounding box precision under bird's eye view
3) 3d: 3D detection bounding box precision
4) aos: Average Orientation Similarity of object

To further evaluate the improvement may possees by introducing mean teacher paradigm, we decided to using improved step between the original ST3D model and amended one to represent the effectiveness of our work **improved step** = $\frac{AP_{amended\ model} - AP_{source\ only}}{AP_{original\ ST3D} - AP_{sousrce\ only}} \times 100\%$

Implementation Details. The ST3D pipeline we use to evaluate and train uses the PVRCNN [10] detector based on OpenPCDet [6] , including the implementation of Random Object Scaling, Curriculum Data Augmentation and Memory Bank of pseudo labels. For the ROS part, *random object scaling*, *random world flip* along X and Y aixs, *random world rotation*, etc, are used to eliminate the bias over object size on the source domain. The pseudo label in memory bank updates every 4 epoches, with a total of 16 epoches since from last time training we discover that model performance during latter period of training turns out to remain almost same or even drop due to possible pseudo label noises.

4.2 Initial Results

Here we used the model trained with nuScense dataset only, and perform evaluation and training on the source only model. With the help of mean teacher paradigm, the average precision of criterion (0.7, 0.5, 0.5) now achieves improvements up to around 10, with the hard tasks improving noticeably, *the digits after second plus sign in each paranthesis represents the improvements results from mean teacher paradigm*,

Car(AP_R11)	0.7(Easy)	0.5(Moderate)	0.5(Hard)
bbox	95.84(+9.58+0.9)	87.39(+16.79+2.3)	87.24(+10.65+8.1)
bev	96.14(+8.47+0.6)	90.78(+11.20+2.4)	91.44(+10.67+3.0)
3d	96.09(+8.53+0.8)	88.18(+11.80+0.4)	88.13(+11.4+0.4)
aos	95.73(+9.27+1.8)	87.0(+16.08+3.0)	86.67(+10.49+8.3)
Car(AP_R40)	**0.7(Easy)**	**0.5(Moderate)**	**0.5(Hard)**
bbox	97.48(+10.28+1.0)	90.13(+14.40+5.2)	90.12(+13.71+7.0)
bev	97.86(+7.74+0.2)	93.76(+14.50+0.4)	93.92(+13.95+0.3)
3d	97.71(+7.83+0.4)	92.75(+14.30+2.0)	92.87(+13.78+2.0)
aos	97.35(+9.91+1.6)	89.63(+14.21+5.5)	89.48(+13.62+7.2)

but the mean teacher paradigm has its most power when dealing with the more strict criterion (0.7, 0.7, 0.7) and especially the 3d evaluation, with up to around 30 improvements in the average precision, which is a really astonishing achievements.

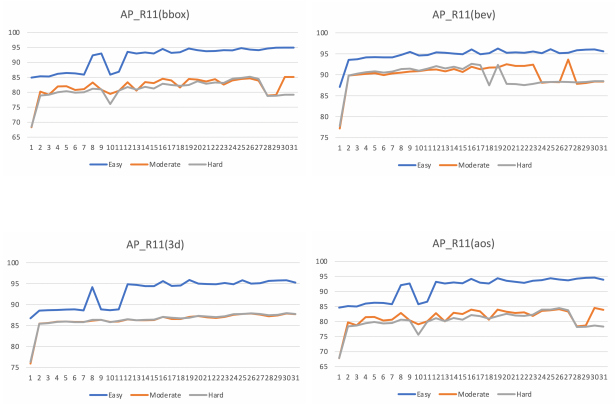
Car(AP_R11)	0.7(Easy)	0.7(Moderate)	0.7(Hard)
bbox	94.48(+9.58)	85.98(+16.34+1.31)	86.08(+10.51+7.04)
bev	88.06(+11.73+2.51)	83.05(+11.38+2.52)	78.13(+11.36+3.4)
3d	76.56(+8.52+24.39)	69.53(+5.17+27.23)	65.36(+6.8+23.8)
aos	94.35(+9.27+0.44)	85.4(+16.08+1.52)	85.31(+10.49+6.97)
Car(AP_R40)	**0.7(Easy)**	**0.7(Moderate)**	**0.7(Hard)**
bbox	96.56(+10.27+0.13)	87.54(+14.4+2.64)	87.71(+13.71+4.57)
bev	92.51(+9.8+6.3)	83.23(+13.85+5.86)	81.92(+13.25+6.49)
3d	77.02(+7.9+27.41)	68.48(+7.66+27.58)	67.41(+6.97+29.0)
aos	96.43(+10.55+0.64)	86.93(+14.21+2.84)	86.87(+13.62+4.58)

Also we can see that other tasks apart from 3d also improves aroud 5 to 10 in average precision.

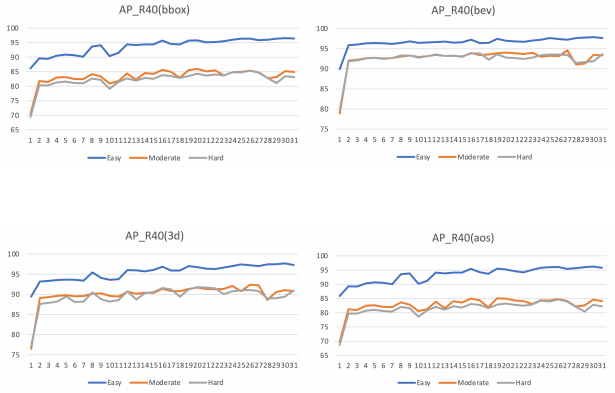
4.3 Epoch-level evalution (ST3D)

Since the overall training is divided into 30 epochs, and for each epoch a check point is generated for further analysis. The evaluation of each epoch is performed and shows some features that leads to our ideas.

The graph below shows the AP from the source only model through out 30 epoch to the final model output, with epoch 1 being the source only model. And corresponding to the four evalution metrics, namely bbox, bev, 3d and aos.



The evalution based on R40 metrics are also shown below, with a similar results,



From the above graph, apart from the AP increase more information reveals the training details of the model. First noticeably feature is the turbulence around the training process, with the AP curver oscillating for 5%. Also from the evalution of bbox, 3d and aos there exists a accuracy sudden drop around epoch 10. Together with the step increase around first batch and possible overfitting around lather epochs. These features give us some idea of the improvement of the ST3D pipeline.

4.4 Analysis (ST3D)

The gap between increase of the easy and moderate & hard tasks may due to the reason that the easy tasks already achieves a rather high average precision while the moderate and hard tasks still have a considerable way to increase.

And since the four evalution metrics are closely related to the training in source domain, it's no wonder that they have the similar evalution curver.

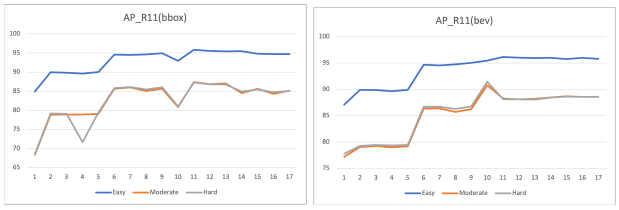
As for the sudden performance drop, one reckon is that this is due to the difference between the LiDAR scanning precision, with nuScenes containing point cloud data of 32 lines and KITTI uses 64 lines, some parts of velodyne data is loss due to less precise training.

With the turbulence of AP during the training process, together with the possible overfitting in the latter stages of unsupervised adaptation, the possible reason that comes first is the pseudo label noises. As mentioned in the last report, generating labels purely rely on the unlabeled target domain may involves with multiple uncertainties, including *Classification noise* and *Localization noise*. Though the extra memory bank is applied to eliminate some of them by selection and voting to filter the better quality ones, the problem is particular obvious around the former epochs, when the memory bank doesn't have abundant pseudo labels for voting. And the low quality labels generate from memory bank are feed into the model again, together with augmentation may enlarge the noises and cause oscillating. And the Mean teacher paradigm is suitable solution for eliminate noises. With the teacher model replacing the unstable student model during the training process, we expect to have a smoother accuracy curver around the training process and possibly higher final model evalution.

4.5 Epoch-level evalution (Mean-teacher)

Since the overall training is divided into 16 epochs, and for each epoch a check point is generated for further analysis. The evaluation of each epoch is performed and to reveal the training process.

The graph below shows the AP from the source only model through out 16 epoches to the final model output, with epoch 1 being the source only model. And corresponding to the four evalution metrics, namely bbox, bev, 3d and aos.

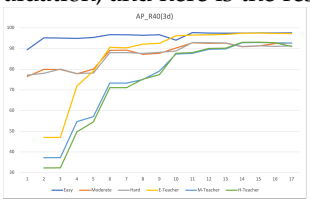


The evalution based on R40 metrics are also shown below, with a similar results,

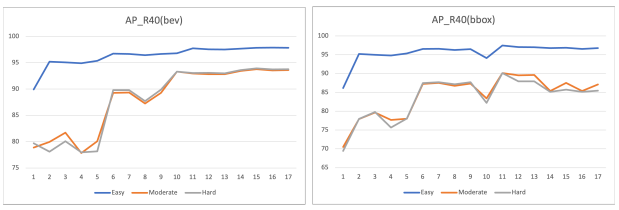
From the above graph, apart from the AP increase more information reveals the training details of the model. First noticeably feature is the turbulence of performance gets smaller, with a smoother curve and higher final average precision. Though their still exists some performance oscillation, but since it's the student model which used for EMA updates, it's fine for them to reveal some of the trend model goes. And thanks to the less pseudo label noise, the model experience less with overfitting or performance drop comparing to the raw ST3D pipeline which performance of model may drop as slightly noise in pseudo label feeds into the system and amplified afterwards.

4.6 Teacher-student evaluation

To perform the comparsion between teacher model and student model to see how them interact and impact each other, we use the most represnetative 3d tasks under the R40 precision evaluation, and here is the result.



Due to the changes of the alpha in EMA updates, the teacher model at first reacts less repsonsively to the improvements brought by the training. But as epoches steps, teacher model become higher and closer to student model, this is when the teacher model really workes, by restrict the pseudo label noise from generating noisy label into the further training. With the restriction of teacher model, the student model experience much less with overfitting and latter period performance drop, which may largely impacts the final result of training.



4.7 Analysis & Progress

In the first try, we add the mean teacher updates into iteration of each epoch training, and we noticed the significant performance drop after 4^{th} epoch. Reminding that the pseudo label is generated every 4 epoches, and here we use teacher model to generates the pseudo label. Then we speculate that maybe is the slow pace updating of teacher model that didn't catch up with the rapid progress student model, and therefore generating pseudo label with less quality and affect the performance.

After reading more about implementation of mean teacher EMA updates, we noticed that many of the implemenation during training involves variable alpha which may change through the training process, with alpha being large during former stages to sync the rapid improvement of student model, and gradually decrease smaller as training enters latter stages, with alpha being small and filter noises brought by pseudo labels. Therefore we improve the mean teacher by decrease alpha from 1 to 0.01 linearly to the epoch number and use fixed alpha for single epoch during every iteration in it. And after this another round of training is performed and evaluated to get the final result we have now.

As it's explained above that domain adaptation problem often has great improvement during the former stage, for its large improvements the model can get and with minor training the rather high results it can get. But as the training goes into further, the improvements may gets little and little, with other noises comes in and overfitting shown up. This is the time when mean teacher works, by filtering the noises and restrict the unwanted oscillation, both the model can progress with small steps but stable paces.

5 Future Plan

As the mean teacher paradigm finally adds to the ST3D pipeline and show its capabilities during the training process, we are thrillled to see the final results as rather small improvements during the latter stage means a lot. And we also witness significant improvements in 3d detection. If we procceed to this topic, we are planning to explore more of the technicals in the unsupervised domain adaptation and by combining some of the technics to construct a pipeline of ourselves.

References

[1] Y. Wang, X. Chen, Y. You, L. Erran, B. Hariharan, M. Campbell, K. Q. Weinberger, and W.-L. Chao, "Train in germany, test in the usa: Making 3d object detectors generalize," 2020. [Online]. Available: https://arxiv.org/abs/2005.08139

[2] Y. Chen, W. Li, C. Sakaridis, D. Dai, and L. Van Gool, "Domain adaptive faster r-cnn for object detection in the wild," in *2018 IEEE/CVF CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR)*, ser. IEEE Conference on Computer Vision and Pattern Recognition. IEEE; CVF; IEEE Comp Soc, 2018, pp. 3339–3348, 31st IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, JUN 18-23, 2018.

[3] K. Saito, Y. Ushiku, T. Harada, and K. Saenko, "Strong-weak distribution alignment for adaptive object detection," in *2019 IEEE/CVF CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR 2019)*, ser. IEEE Conference on Computer Vision and Pattern Recognition. IEEE; CVF; IEEE Comp Soc, 2019, pp. 6949–6958, iEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, JUN 16-20, 2019.

[4] Y. Zheng, D. Huang, S. Liu, and Y. Wang, "Cross-domain object detection through coarse-to-fine feature adaptation," 2020. [Online]. Available: https://arxiv.org/abs/2003.10275

[5] J. Yang, S. Shi, Z. Wang, H. Li, and X. Qi, "St3d: Self-training for unsupervised domain adaptation on 3d object detection," in *2021 IEEE/CVF CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, CVPR 2021*, ser. IEEE Conference on Computer Vision and Pattern Recognition. IEEE; IEEE Comp Soc; CVF, 2021, pp. 10 363–10 373, iEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), ELECTR NETWORK, JUN 19-25, 2021.

[6] J. Deng, S. Shi, P. Li, W. Zhou, Y. Zhang, and H. Li, "Voxel r-cnn: Towards high performance voxel-based 3d object detection," 2020. [Online]. Available: https://arxiv.org/abs/2012.15712

[7] Z. Luo, Z. Cai, C. Zhou, G. Zhang, H. Zhao, S. Yi, S. Lu, H. Li, S. Zhang, and Z. Liu, "Unsupervised domain adaptive 3d detection with multi-level consistency," 2021. [Online]. Available: https://arxiv.org/abs/2107.11355

[8] 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," 2019. [Online]. Available: https://arxiv.org/abs/1903.11027

[9] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354–3361.

[10] S. 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," 2019. [Online]. Available: https://arxiv.org/abs/1912.13192