Leveraging HD Maps for 3d Object Detection

Sreehari Sreejith
Georgia Tech
ssreejith3@gatech.edu

Shalini Chaudhuri
Georgia Tech
shalini.chaudhuri@gatech.edu

Abstract

In our project, we explore if encoding information from HD Maps enhances 3d object detection for Autonomous Driving. Objects in a 3D world do not follow any particular orientation, and box-based detectors have difficulties enumerating all orientations or fitting an axis-aligned bounding box to rotated objects. CenterPoint [14] is a new state of the art[1] *3d object detection and tracking technique that uses keypoint features to address the same. It predicts object centers and then regresses to other attributes such as 3d size, orientation and velocity (for tracking). CenterPoint is both simple and performant! In our work, we explore if we can better inform this approach to detection, by using a new signal generated from HD map data that is readily available in AV datasets like Argoverse [2] and NuScenes [1]. We experiment with a few approaches and find that using additional map information does show promise in helping improve 3d object detection performance.*

1. Introduction

Most autonomous driving systems rely on 3D perception to enable navigation and motion on the road. 3D object detection forms a core part of the stack that solves perception and prediction problems for the self driving vehicle. 3D object detection allows the system to capture information about the size, orientation and position of the objects in the scene, all of which are integral to downstream applications such as motion prediction and planning.

3D object detection on LiDAR point clouds is especially challenging, given the sparsity of the data at long ranges. Translating 2D detection methods to 3D detection methods is non-trivial due to the difference in the properties of data, therefore, new methods to solve this problem are necessary. CenterPoint[14] overcomes the challenges of 3D object detection by representing objects as points in space. CenterPoint uses PointPillars [6] as a backbone network to learn the representation of point clouds in 3D space. It then uses a kepyoint detector to predict the center of each object in the scene and finally regressing to other attributes like size, orientation and velocity.

In our experiments, we propose to enhance the performance of CenterPoint by augmenting the encoded point clouds with information from HDMaps of the scene. We believe that HD maps should be a good signal to use for 2 reasons: (1) We are typically interested in capturing objects in and around the drivable area and thus any semantic information from the HD Maps regarding this would be a useful signal; (2) This signal should better inform the model about the kind of constraints that would naturally apply to a detected 3d object in terms of its orientation and bounding box. For eg., given lane markers and a car driving within a lane, it should be easier to predict the likely bounding box and orientation using the same.

2. Related Work

Early works in 3D object detection focused on single modalities such as monocular images [3] but they suffer from problems with depth estimations. More recent work focuses on utilizing data from LiDAR sensors, VoxelNet [16] divides the 3D space into equally spaced voxels and aggregates the information inside each voxel. PIXOR [13] speeds up the computation by removing 3D convolutions in space and by encoding information in the features of the 2D feature map. PointPillars [6] uses longitudinal pillars of point clouds instead of voxels and improves the efficiency of the backbone network.

VoteNet [8] attempts to solve 3D object detection by using Hough voting on learned point features. [11] utilize multiple points in the point anchors to regress other attributes. [14] uses a single point (object center) to regress attributes for 3D object detection.

Other recent works such as [15] utilize maps generated online to reason better the location of objects in urban scenarios. HDNet[12] use HD Maps to exploit priors that can help improve the performance of the object detector.

[1]SOTA on the nuScenes leaderboard at the time of our project proposal

3. CenterPoint architecture

Our work modifies the CenterPoint model [14] to include map features to perform 3d object detection. The CenterPoint architecture is show in Figure 1. We use an instance of CenterPoint which uses a PointPillar [6] 3d backbone as our reference architecture. The 2d backbone that is shown in the figure is the RPN layer [9], the output of which is 128x128 feature maps that gets passed to the CenterHead (referenced as 'Head') in the image.

In our experiments, we modify the architecture by passing in additional channel-wise concatenated map features to the CenterHead as shown in Figure 2 followed by network training/finetuning. For instance, a straightforward way to do this is by simply passing HD maps that are resized to the same dimensions (i.e., raw 128x128 RGB pixel values) as the 2d backbone output. We detail the specific approaches we explore to generate these features in the section below.

4. Experiments

The main theme around all our experiments was to explore how we can enable the CenterHead of CenterPoint to be able to use these map features before it computes the center and bounding box values along with orientation and velocity. We conduct the following broad sets of experiments to incorporate map information as an additional signal for 3d object detection.

1. Direct concatenation of raw HD map images
2. ResNet-based [5] Map feature extractor network
3. UNet-based [10] Map feature extractor network
4. Direct concatenation of semantic map images

In addition to the above experiments, to investigate some gaps in results between the pretrained baseline and our experiments initialised using the pretrained CenterPoint model, we run an additional experiment to show that the differences are not due to our current design choices, but might have more to do with other factors such as the size and distribution of our dataset (which is a small subset of the entire nuScenes dataset), training setting under a single GPU instance, etc.

4.1. Dataset and Metrics

4.1.1 Dataset

We use the nuScenes dataset for all our experiments. Due to limited availability of resources[2], we stick to using the official nuScenes mini dataset which comprises approximately 400 samples (i.e., frames containing bounding box annotations) to train on and around 80 samples for validation.

[2]We use a single GCP GPU instance using the limited free credits available to students

4.1.2 Metrics Used

We use the official metrics defined and used on the nuScenes [1] leaderboard[3]. The main metric used is mean Average Precision (mAP). Here, the mean average precision is calculated by considering the match using BEV 2d center distance and integrating the precision-recall curve for precision and recall greater than 0.1. The average over match thresholds of 0.5m, 1m, 2m and 4m (m=meters) is taken for the final mAP value.

The other key metrics that are used include: (definitions below are borrowed from the nuScenes [1] website):

1. ATE - Average Translation Error: This is the difference in euclidean center distance in meters (unbounded)
2. ASE - Average Scale Error: Scale error is calculated as 1-IOU after aligning centers and orientation
3. AOE - Average Orientation Error: AOE is the smallest yaw angle difference between prediction and ground truth in radians
4. AVE - Average Velocity Error: This is the absolute velocity error in m/s when applicable (unbounded)
5. AAE - Average Attribute Error : This is calculated as 1-acc where acc is the attribute classification accuracy.

Using the above, the nuScenes detection score, i.e., NDS metric is computed as follows: the errors are first converted to scores and (using $score = max(1 - error, 0)$) and then performing a weighted sum using a weight of 5 for mAP and 1 for the other scores.

4.2. Direct Concatenation of HD Maps

To test our original hypothesis, we decided to use the rasterized BEV image of the HDMap and concatenate it directly channel-wise with the encoded information from the point clouds. This is shown in Figure 2.

We perform 2 experiments ($E1$ to $E2$) with this set-up.

1. $E1$: Pretrained CenterPoint model with map passed as an additional input. For this experiment we pass in HD map images of size 128x128 to the CenterHead.
2. $E2$: CenterPoint model is trained from scratch with map passed as an additional input. For this experiment we pass in HD map images of size 128x128 to the CenterHead.

[3]https://www.nuscenes.org/object-detection/?externalData=allmapData=all

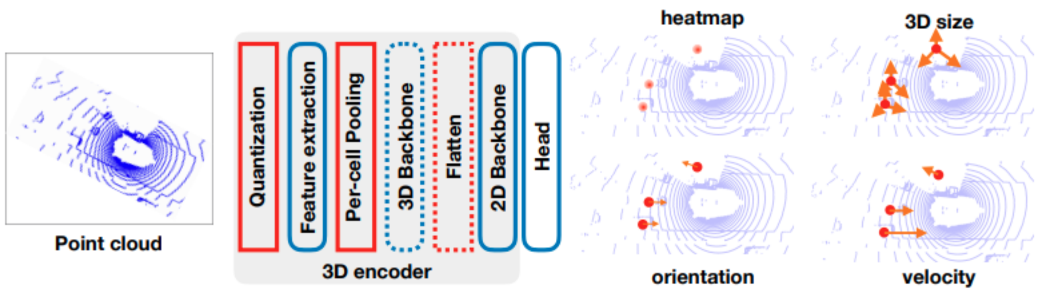


Figure 1: CenterPoint model - We use an instance of CenterPoint that uses a PointPillar 3d backbone as our reference architecture that we modify in order to include the map features in all our experiments

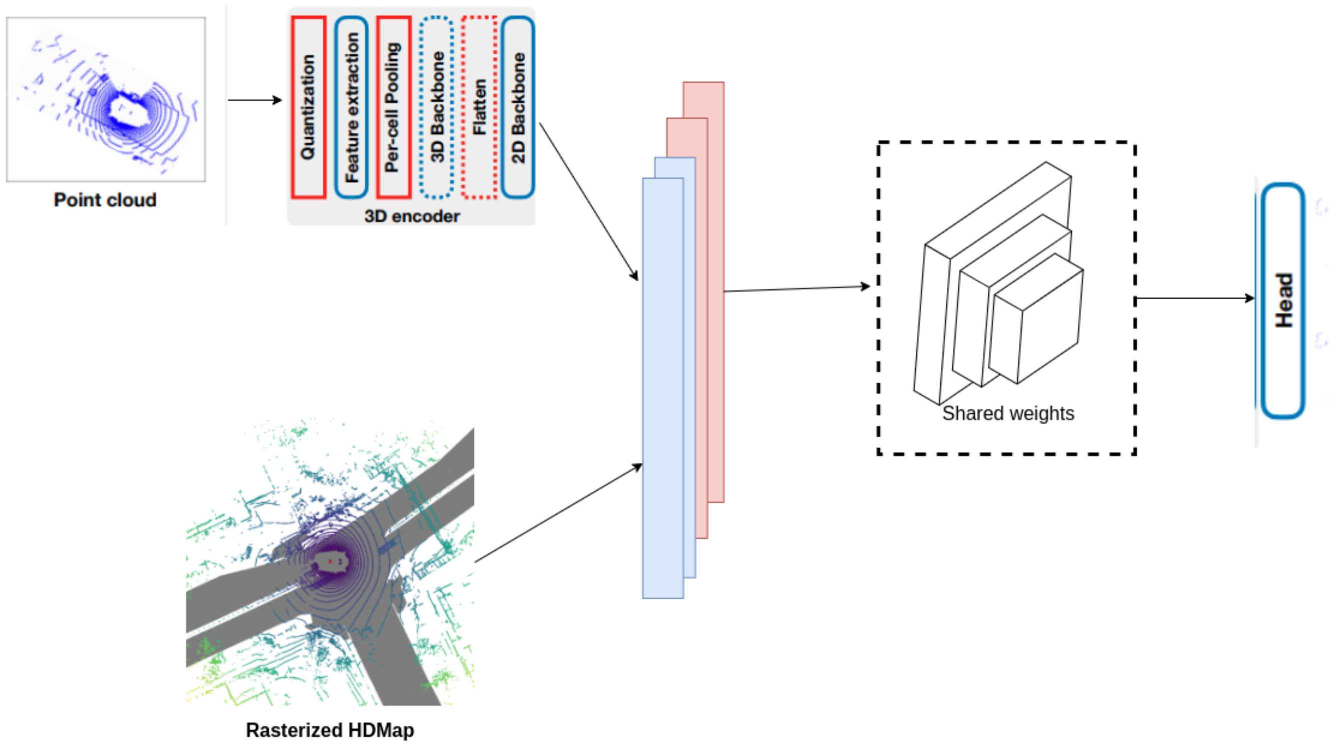


Figure 2: The main theme in our approach is to pass additional map information to the CenterHead of CenterPoint. This additional map information is provided as additional channels with the same spatial dimensions as that of the 2d backbone output and it flows through a shared conv layer followed by the individual regression heads that CenterPoint uses.

4.3. ResNet-based Map feature extractor network

ResNet [5] is a deep residual network that has been effective at solving computer vision problems due to its expressiveness and ability to deal with the vanishing gradient problem. We want to extract useful information from the HD Maps and feed it to the CenterHead. By extracting better map features, the network can learn constraints such as, pedestrians only occur on walkways and intersections, and so on. Therefore, having a network that can learn low level map features from our HD Maps would enhance the performance of our 3D object detector. We perform 4 experiments ($R1$ to $R4$). For all four, we use a pretrained (on ImageNet [4]) ResNet50 as a feature extractor for HDMaps,

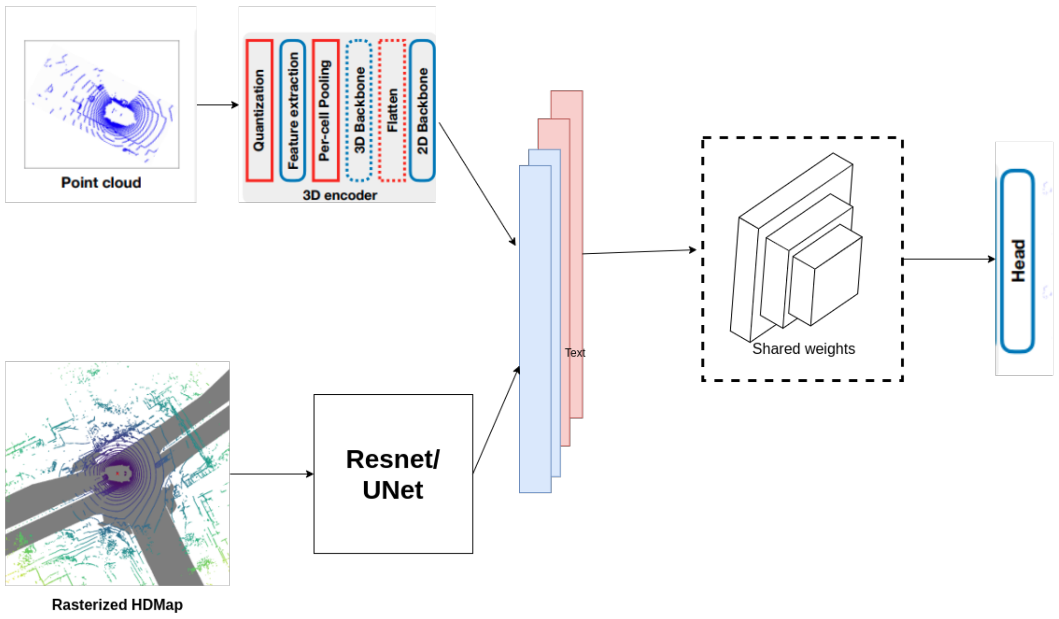


Figure 3: ResNet/UNet map feature extractors - We experiment with using Resnet and UNet models as backbone map feature extractor networks to extract map features that are channel-wise concatenated to the outputs of the 2d backbone from CenterPoint

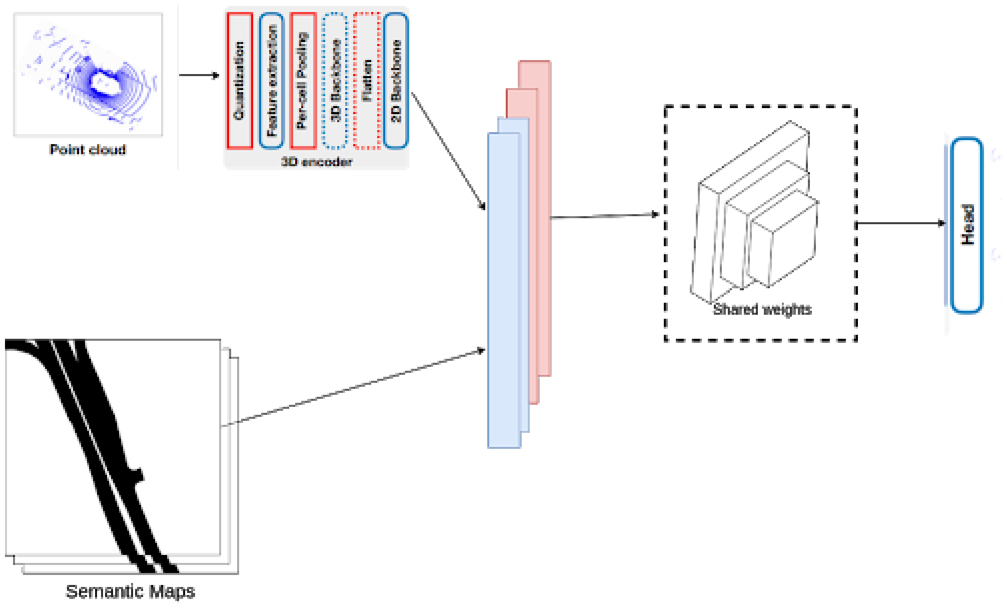


Figure 4: Semantic maps experiment - additional map information is provided by concatenating semantic maps having the same spatial dimensions as that of the 2d backbone output. Here, we replace the single rasterized HD map with semantic maps, which are binary masks indicating drivable area, stop lines, etc.

as shown in Figure 3. For experiments $R1$ and $R2$, we use the pretrained CenterPoint model to load and freeze weights upto the 3d encoder. For experiments $R3$ and $R4$, we train the entire CenterPoint model from scratch. In all experiments, we use the pretrained ResNet weights to initialize the map feature extractor and also train the CenterHead (i.e., the shared conv layer alongwith the regression heads) from scratch.

1. $R1$: Pretrained CenterPoint model with map features generated by the pretrained ResNet block (i.e., Resnet weights are frozen during training). For this experiment we pass in HD map images of size 1024x1024 and extract map features of size 128X128 from the map backbone network.

2. $R2$: Pretrained CenterPoint model with map features generated by the finetuned ResNet block. For this experiment we pass in HD map images of size 1024X1024 and extract map features of size 128X128 from the map backbone network.

3. $R3$: CenterPoint model is trained from scratch with the map features generated by the pretrained ResNet block. For this experiment we pass in HD map images of size 1024x1024 and extract map features of size 128X128 from the map backbone network.

4. $R4$: CenterPoint model is trained from scratch with the map features generated by the finetuned ResNet block. For this experiment we pass in HD map images of size 1024X1024 and extract map features of size 128X128 from the map backbone network.

4.4. UNet-based Map feature extractor network

UNet [10], which is an extension over the fully convolutional network [7] is a model that was proposed for biomedical image segmentation. The ideal information we would want to pass to the CenterHead is a notion of what each pixel in the map represents (or is a part of), in order for the network to be able to learn things that humans understand intuitively, for eg., cars tend to be parked a certain way with respect to the drivable area and footpath, etc. Intuitively, one way to convey that information to the network, could be to supplement the network with features providing the semantic segmentation information (Note: we try using semantic maps directly using the map expansion API of nuScenes in a separate experiment). Thus, we decided to finetune a UNet model to learn map features that are relevant for 3d object detection.

We perform 4 experiments ($U1$ to $U4$). For all four, we use a ResNet based encoder for the UNet model and choose between 2 or 4 decoder blocks based on the experiment. For experiments $U1$ and $U2$, we use the pretrained centerpoint model to load and freeze weights upto the 3d encoder. For experiments $U3$ and $U4$, we train the entire CenterPoint model from scratch. In all experiments, we use the pretrained ResNet weights to initialize the UNet encoder and also train the CenterHead (i.e., the shared conv layer alongwith the regression heads) from scratch.

1. $U1$: Pretrained CenterPoint model with map features generated by the 2nd decoder block. For this experiment we pass in HD map images of size 1024x1024 and extract 128x128 features from the 2nd decoder block.

2. $U2$: Pretrained CenterPoint model with map features generated by the final (4th) decoder block. For this experiment we pass in HD map images of size 128x128 and then extract 128x128 features from the final decoder block.

3. $U3$: CenterPoint model is trained from scratch with the map features generated by the 2nd decoder block. For this experiment we pass in HD map images of size 1024x1024 and extract 128x128 features from the 2nd decoder block.

4. $U4$: CenterPoint model is trained from scratch with the map features generated by the final (4th) decoder block. For this experiment we pass in HD map images of size 128x128 and then extract 128x128 features from the final decoder block.

4.5. Direct concatenation of semantic map images

Instead of using a single rasterized image of the HD Map, we use multiple semantic maps, i.e., binary masks indicating the drivable area, stop lines, walkways, intersections and lanes in the scene. The intuition behind splitting up the HD Map into multiple components is to enable our network to learn smaller and simpler signals from the data. It allows the network to constrain the regression head better.

We conduct 2 experiments ($S1$ to $S2$) with semantic maps passed as additional inputs to the CenterHead.

1. $S1$: Pretrained CenterPoint model with 7 semantic maps concatenated channel-wise to the CenterHead input. For this experiment we pass in images of size 128x128 to the CenterHead.

2. $S2$: CenterPoint model with 7 semantic maps concatenated channel-wise to the CenterHead input. For this experiment we pass in images of size 128x128 to the CenterHead.

4.6. Additional: Finetuning CenterHead

One of the observations in the results of our experiments was that the comparisons to the pretrained baseline centerpoint model weren't showing similar promising results that we were seeing when we trained the model from scratch. There could potentially be many reasons for such results, including but not limited to the following - the centerhead being trained entirely from scratch, the difference in dataset distribution and size between our mini dataset and the full nuScenes dataset comprising 1000 scenes and even the training setting (i.e., use of smaller batch sizes on a single GPU vs distributed training on larger batch sizes). We wanted to confirm if training the CenterHead from scratch is a valid design choice and whether that was causing the difference in results and in order to eliminate it from the potential list of causes for the poor numbers. For the other factors, the only way to experiment was to expand our training dataset (which was not feasible due to our resource constraints). For this, we ran a couple of experiments where we finetuned the CenterHead along with the map network. Note that in the experiments described, we have always learned these CenterHead weights from scratch, and so, this experiment verifies if the difference observed is due to the same.

Here, we load pretrained weights on the full network, but fine-tune only the CenterHead (i.e., the shared conv layer and the regression heads) and map net. We run 2 experiments - In $F1$, the HD map images are channel wise concatenated to the RPN output (Experiment-F1). In $F2$, the map features are generated by the UNet model from the final decoder block (Experiment-F2).

5. Results

The results for our first set of experiments, i.e., Direct concatenation of raw HD map images is shown Table 1. The results seen here are quite encouraging as we note that there's minor improvements to the mAP and avg translation error and importantly also to the avg orientation error. This was something we intuitively expected to see with the use of HD Maps because the HD map information should come in handy in learning appropriate constraints (for instance, with respect to the orientation) to the regression head to make better predictions more consistently.

For the second and third set of experiments, where we use ResNet and UNet as map feature extractors, the results are shown in tables 2 and 3 respectively. While using ResNet as the map feature extractor (Table 2), one observation from our results is that using a frozen resnet as feature extractor gives us slightly better mAP in both experiments, but the finetuned ResNet performs marginally better on the other metrics. Second, we see that there are minor improvements to performance when we retrain the CenterPoint model. But given that ResNet is pretrained on ImageNet, it's likely that the domain shift in going from ImageNet to HD Maps needs to be explicitly addressed if we are to expect better features to be extracted and in order for us to see better improvements.

When UNet is used as the map feature extractor (Table 3), it again shows minor improvements in performance which is consistent across both mAP and the other scores as well. It is also interesting to see that between using map features from the 2nd and final decoder blocks, the latter seems to consistently perform better. This means that our intuition behind using the UNet model and it's symmetricity might be useful to extract relevant map features. Overall, we see that using Resnet or UNet as feature extractors shows similar minor improvements and the results are quite promising because the network is indeed learning to extract useful information for the final regression.

In the results discussed above, we note that when comparing to baseline1, there is a significant gap between the performance of our models and the one trained by the authors. In order to address this, we perform an additional experiment to verify if this is being caused due to our design choice wherein we train the CenterHead from scratch. Based on the results shared in Table 5, we see that this is not necessarily the case as there is still a significant gap between our models and baseline-1 even with the CenterHead being finetuned after being initialized using pretrained weights. While, we see that there is some improvement to the mAOE, mAVE and mAAE metrics, the mAP does not improve. This experiment tells us that there are other factors at play - potentially, the size and distribution of the mini-dataset versus that of the entire nuScenes dataset, the training setting including number of GPUs and batch size (which are reduced in our case due to limited resource availability) - which might be bigger factors behind why these numbers don't come close to each other in spite of them using the same pretrained CenterPoint model.

Finally, in Table 4, we see that using semantic maps helps improve our results, and in fact, the increase in improvement seen is more than that shown when we use Resnet/Unet on our rasterized HD maps. This again indicates that passing additional information to the centerhead is indeed useful. Specifically, the improvements here could be due to the fact that the semantic maps are an easier, simpler signal to learn which better constrains the regression head of the network, thus providing better predictions. These results agree quite well with our initial intuitions on how maps might be helpful and also our decision to pick UNet as a feature extractor - in both cases, we were looking to inform the model about the presence of various clues such as driving areas, lanes, etc that could help better constrain object prediction.

We share the per-class mAP values (tables 6, 7, 8, 9 and 10) and a few sample qualitative results from our baselines and the experiments using semantic maps (best-performing) along with representative PR curves in the appendix section. These qualitative results are in-line with the metrics presented here and indicate both successful and failure cases and show that there is a still a lot of room for improvement.

6. Conclusion

Overall, we explored different ideas to incorporate HD Map information in 3d object detection. Specifically, we

Overall Results	**mAP ↑**	**mATE ↓**	**mASE ↓**	**mAOE ↓**	**mAVE ↓**	**mAAE ↓**	**NDS ↑**
Baseline 1 (Pretrained)	.4206	.4515	.4514	.5613	.4347	.3122	.4892
Baseline 2 (Trained from Scratch)	.1125	.6725	.5191	1.1354	2.4062	.7615	.1609
Experiment-E1 (Pretrained Model with HDMaps)	.3260	.4987	.5067	.8786	2.9446	0.7691	.2977
Experiment-E2 (Trained from scratch with HDMaps)	**.1412**	**.6520**	.6076	**.9873**	2.8212	**.7579**	**.1701**

Table 1: Performance of direct concatenation of raw HD map images against baselines. We see that when training the model from scratch, the model using HD map information outperforms baseline-2 in both mAP as well as the mATE and mAOE scores which are in line with our intuition on how maps can contribute to improvements in 3d object perception

Overall Results	**mAP ↑**	**mATE ↓**	**mASE ↓**	**mAOE ↓**	**mAVE ↓**	**mAAE ↓**	**NDS ↑**
Baseline-1 (Pretrained)	.4206	.4515	.4514	.5613	.4347	.3122	.4892
Experiment-R1 (Pretrained CP and pretrained RN)	.3242	.5393	.5250	.8613	3.8923	.7731	.2922
Experiment-R2 (Pretrained CP and finetuned RN)	.3089	.5265	.4911	.8445	3.2784	.7487	.2933
Baseline-2 (Trained from Scratch)	**.1125**	**.6725**	.5191	**1.1354**	2.4062	**.7615**	**.1609**
Experiment-R3 (CP trained from scratch and pretrained RN)	**.1222**	.7103	.5461	1.2391	6.9995	.7644	.1590
Experiment-R4 (CP trained from scratch and finetuned RN)	**.1111**	**.6399**	.5421	1.1456	2.4399	**.7716**	**.1602**

Table 2: Results on experiments using a ResNet map feature extractor

experimented with these ideas using the CenterPoint architecture as a reference architecture and explored a few different ways in which we could feed additional map information as a signal to the CenterHead part of the network which is responsible for the final regression. Our experiments included the direct use of pixels, a ResNet feature extractor, a UNet feature extractor and exploring the use of semantic maps. We found that using HD Maps as an additional signal does show minor improvements in performance, based on our experiments where we train the model from scratch. Given that the improvements are shown on the mini dataset, we note that they aren't necessarily conclusive and require further training and evaluation using the larger nuScenes dataset. For future work, we hope to validate our findings on the larger dataset and also explore the option of combining semantic maps with the rasterized HD Map image to extract better map features.

References

[1] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11621–11631, 2020.

[2] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8748–8757, 2019.

[3] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun. Monocular 3d object detection for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[6] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. Pointpillars: Fast encoders for object detection from point clouds. *CVPR*, 2019.

[7] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[8] C. R. Qi, O. Litany, K. He, and L. J. Guibas. Deep hough voting for 3d object detection in point clouds, 2019.

Overall Results	mAP ↑	mATE ↓	mASE ↓	mAOE ↓	mAVE ↓	mAAE ↓	NDS ↑
Baseline-1 (Pretrained)	.4206	.4515	.4514	.5613	.4347	.3122	.4892
Experiment-U1 (Pretrained CP; UNet map features from 2nd decoder block)	.3023	.5105	.4957	.8687	2.5269	.7696	.2867
Experiment-U2 (Pretrained CP; UNet map features extracted from final decoder block)	.3048	.5057	.4903	.8140	2.4838	.7652	.2949
Baseline-2 (Trained from Scratch)	**.1125**	**.6725**	.5191	**1.1354**	2.4062	**.7615**	**.1609**
Experiment-U3 (CP trained from scratch; UNet map features extracted from 2nd decoder block)	**.1221**	.6554	.5253	1.2155	2.4353	**.7727**	**.1657**
Experiment-U4 (CP trained from scratch; UNet map features extracted from final decoder layer)	**.1155**	**.6317**	.5282	1.1842	**1.9566**	**.7398**	**.1678**

Table 3: Results on experiments using a UNet map feature extractor

Overall Results	mAP ↑	mATE ↓	mASE ↓	mAOE ↓	mAVE ↓	mAAE ↓	NDS ↑
Baseline-1 (Pretrained)	.4206	.4515	.4514	.5613	.4347	.3122	.4892
Experiment-S1 (Pretrained Model with semantic maps)	.3108	.5149	.4964	.8656	3.7633	.7595	.2918
Baseline-2 (Trained from Scratch)	.1125	.6725	.5191	1.1354	2.4062	.7615	.1609
Experiment-S2 (Trained from scratch with semantic maps)	**.1295**	**.5868**	.5280	**1.0453**	6.1381	**.7558**	**.1777**

Table 4: Results using direct concatenation of semantic maps as additional channels to the CenterHead input. In using semantic maps, we see better increase in numbers when compared to using ResNet and UNet feature extractors on the single rasterized map image. Experiment-S2 (i.e., last row) is our best performing model overall in terms of NDS score when we train the model from scratch

[9] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[10] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[11] K. Wong, S. Wang, M. Ren, M. Liang, and R. Urtasun. Identifying unknown instances for autonomous driving, 2019.

[12] B. Yang, M. Liang, and R. Urtasun. Hdnet: Exploiting hd maps for 3d object detection. In *Conference on Robot Learning*, pages 146–155, 2018.

[13] B. Yang, W. Luo, and R. Urtasun. Pixor: Real-time 3d object detection from point clouds, 2019.

[14] T. Yin, X. Zhou, and P. Krähenbühl. Center-based 3d object detection and tracking. *arXiv:2006.11275*, 2020.

[15] H. Zhang, A. Geiger, and R. Urtasun. Understanding high-level semantics by modeling traffic patterns. In *Proceedings of the IEEE international conference on computer vision*, pages 3056–3063, 2013.

[16] Y. Zhou and O. Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. *CVPR*, 2018.

Overall Results	mAP ↑	mATE ↓	mASE ↓	mAOE ↓	mAVE ↓	mAAE ↓	NDS ↑
Baseline 1 (Pretrained CP model)	**0.4206**	**0.4515**	**0.4514**	**0.5613**	**0.4347**	0.3122	**0.4892**
Experiment-E1 (Pretrained CP model where regression heads are learned from scratch + Basic Map Net)	0.3260	0.4987	0.5067	0.8786	2.9446	0.7691	.2977
Experiment-F1 (Pretrained CP network with finetuning of shared conv layer and regression heads + Basic Map Net)	0.3147	0.4834	0.4948	0.7546	0.6754	**0.3049**	0.3860
Experiment-F2 (Pretrained CP network with finetuning of shared conv layer and regression heads + UNet Map Net)	**0.3171**	**0.4946**	0.4886	**0.7618**	0.6590	**0.3132**	**0.3868**

Table 5: Results on our additional experiment where we fine-tune the CenterHead and Map networks to see how they compare against Baseline-1. The results show that there is still a significant gap between the baseline model and our models irrespective of whether the center head weights are learned from scratch or fine-tuned, indicating that the difference in the results are not due to this particular design choice and instead potentially due to the use of a mini-dataset and/or our training criteria.

A. Supplementary - Code Repository and select addtional results

A.1. Code

Our code is currently shared on the internal github repo here: Centerpoint-maps (Note: This is a link to Georgia Tech internal github repository)

A.2. Addition Results

We share the per class mAP values obtained from different experiments in tables 6, 7, 8, 9 and 10 here.

In figures 5 and 6, we share select qualitative results and PR curves from both baselines and our best performing experiment, i.e., using semantic maps.

Per Class Results (mAP)	Car	Truck	Bus	Pedestrian	Bicycle	Motorcycle
Baseline 1 (Pretrained)	**.881**	**.711**	**.983**	**.881**	**.269**	.433
Baseline 2 (Trained from Scratch)	.540	.050	.041	.491	.000	.004
Experiment-E1 (Pretrained with HDMaps)	.812	.556	.986	.828	.000	**.078**
Experiment-E2 (Trained from scratch with HDMaps)	**.526**	**.217**	.181	**.481**	.000	**.006**

Table 6: Per class average precision of direct concatenation of raw HD map images against baselines. We see that when training the model from scratch, the model using HD map information outperforms baseline-2 in identifying cars, trucks, pedestrians and motorcycles.

Per Class Results (mAP)	Car	Truck	Bus	Pedestrian	Bicycle	Motorcycle
Baseline 1 (Pretrained)	**.881**	**.711**	**.983**	**.881**	**.269**	.433
Experiment-R1 (Pretrained CP and pretrained RN)	.821	.571	.936	.842	.000	.072
Experiment-R2 (Pretrained CP and fine-tuned RN)	.814	.398	.968	.821	.000	**.088**
Baseline 2 (Trained from Scratch)	.540	.050	.041	**.491**	.000	.004
Experiment-R3 (CP trained from scratch and pretrained RN)	**.555**	**.122**	**.083**	.455	.000	**.007**
Experiment-R4 (CP trained from scratch and fine-tuned RN)	.541	.040	.035	.486	.000	.010

Table 7: Per class average precision using ResNet as a map feature extractor against baselines. We notice a gain in performance in idenitfying vehicles using pretrained ResNet and training CenterPoint from scratch.

Per Class Results (mAP)	Car	Truck	Bus	Pedestrian	Bicycle	Motorcycle
Baseline 1 (Pretrained)	**.881**	**.711**	**.983**	**.881**	**.269**	.433
Experiment-U1 (Pretrained CP; UNet map features from 2nd decoder block)	.826	.389	.894	.821	.000	.094
Experiment-U2 (Pretrained CP; UNet map features extracted from final decoder block)	.810	.339	**.992**	.815	.000	**.093**
Baseline 2 (Trained from Scratch)	.540	.050	.041	.491	.000	.004
Experiment-U3 (CP trained from scratch; UNet map features extracted from 2nd decoder block	**.542**	**.068**	**.095**	**.507**	.000	**.009**
Experiment-U4 (CP trained from scratch; UNet map features extracted from last decoder layer)	**.543**	**.095**	.032	.474	.000	**.012**

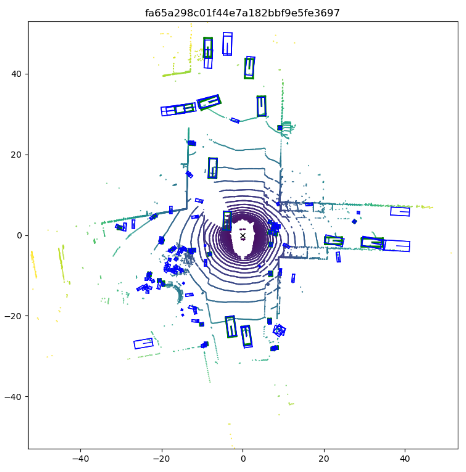
Table 8: Per class average precision using UNet as a map feature extractor against baselines. We notice a gain in performance in idenitfying vehicles and pedestrians using UNet map features and training CenterPoint from scratch.

Per Class Results (mAP)	**Car**	**Truck**	**Bus**	**Pedestrian**	**Bicycle**	**Motorcycle**
Baseline 1 (Pretrained)	**.881**	**.711**	**.983**	**.881**	**.269**	.433
Experiment-S1 (Pretrained Model with semantic maps)	.808	.385	**.987**	.838	.000	.090
Baseline 2 (Trained from Scratch)	.540	.050	.041	.491	.000	**.004**
Experiment-S2 (Trained from scratch with semantic maps)	.523	**.065**	**.246**	.457	.000	**.004**

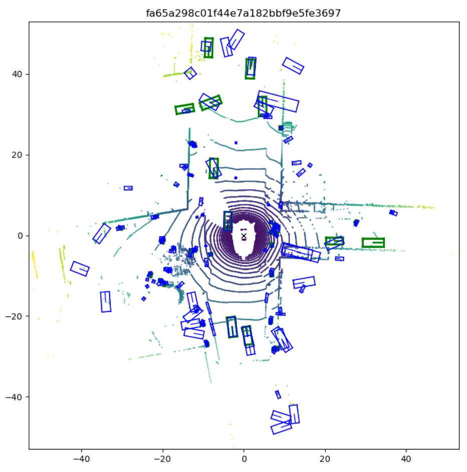
Table 9: Per class average precision of direct concatenation of semantic map masks against baselines. We see improved performance on large vehicles in when training CenterPoint from scratch using semantic maps.

Per Class Results (mAP)	**Car**	**Truck**	**Bus**	**Pedestrian**	**Bicycle**	**Motorcycle**
Baseline 1 (Pretrained CP model)	**.881**	**.711**	**.983**	**.881**	**.269**	.433
Experiment-E1 (Pretrained CP model where regression heads are learned from scratch + Basic Map Net)	.812	.556	.986	.828	.000	.078
Experiment-F1 (Pretrained CP network with finetuning of shared conv layer and regression heads + Basic Map Net)	.837	.417	.971	.833	.000	.088
Experiment-F2 (Pretrained CP network with finetuning of shared conv layer and regression heads + UNet Map Net)	.828	**.422**	**.989**	.831	.000	**.102**

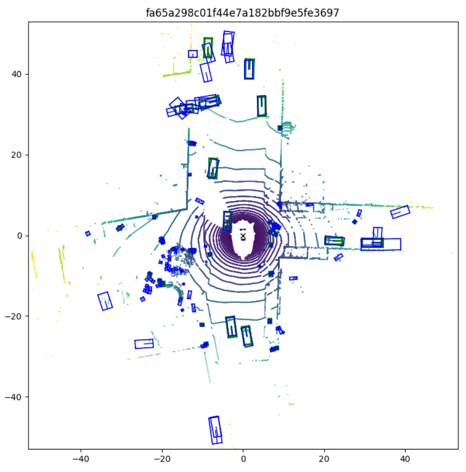
Table 10: Per class average precision of fine-tuned CenterHead and Map networks against Baseline-1.



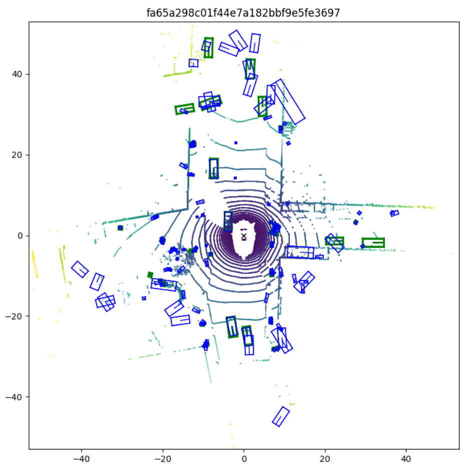
(a)



(b)

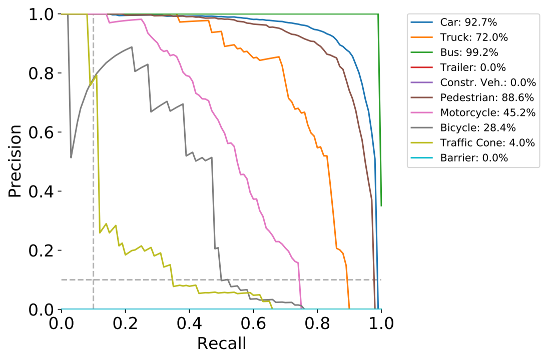


(c)

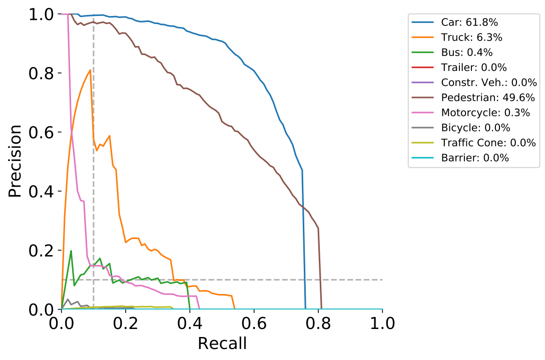


(d)

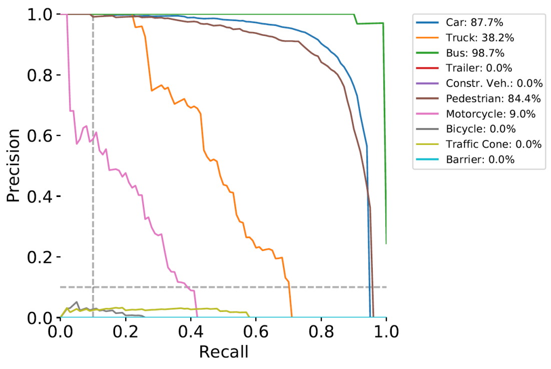
Figure 5: Qualitative results on the same frame from (a) Baseline1 (b) Baseline2 (c) Experiment-S1 (d) Experiment-S2. Blue bounding boxes show us predictions and green gives us the Ground Truth annotation.



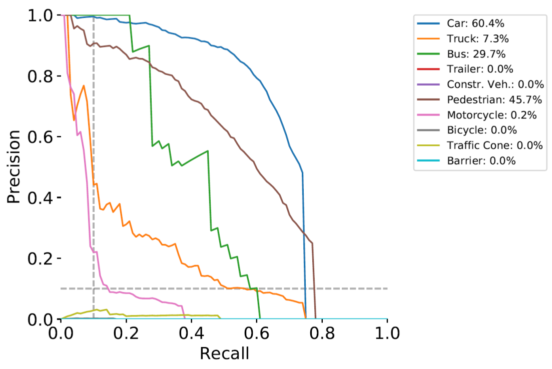
(a)



(b)



(c)



(d)

Figure 6: Precision-Recall curves for (a) Baseline1 (b) Baseline2 (c) Experiment-S1 (d) Experiment-S2. The PR curves are for a threshold of 2.0m.