3D Object Detection and Tracking using Graph Neural Networks on Waymo Dataset

Ahmed Bahnasy
Technical University in Munich
ahmed.bahnasy@tum.de

Abstract

Recent approaches for 3D object detection have made a tremendous progress due to the development of deep learning. this also highly influences the progress in other tasks like 3D Multi-object tracking (MOT) which is an indispensable component to any autonomous system. Recent work uses a standard tracking- by-detection pipeline, where feature extraction is first per- formed independently for each object in order to compute an affinity matrix. Then the affinity matrix is passed to the Hungarian algorithm for data association. This matching can be seen as discrete optimisation process and there is no learning it. A key factor of this standard pipeline is to learn discriminative features for different objects, by relying on a robust detector in order to reduce confusion during data association. In this task, we experiment with improving the discriminative feature learning for MOT: by first build a robust object detector and second instead of obtaining features for each object independently, we try applying learning module for tracking by introducing the Graph Neural Network which enables feature interaction mechanism. As a result, the feature of one object is informed of the features of other objects so that the object feature can lean towards the object with similar feature (i.e., object probably with a same ID) and deviate from objects with dissimilar features (i.e., object prob- ably with different IDs), leading to a more discriminative feature for each object

1. Introduction

Robust 3D perception becomes an essential component in any state-of-the-art autonomous system. 3D detection has a series of challenges: First, point-clouds are sparse, the points are unevenly distributed across the 3D space, and most of the 3D space are without measurements. Second, 3D objects have a wide range of sizes, shapes and aspect ratios, for example, in autonomous driving domain, buses, trucks and cars are elongated, pedestrians are tall and cyclists are nearly planer. These challenges makes applying ideas from 2D domain to 3D domain not straight forward. trying to use the anchor based approach followed in 2D domain by assigning a different anchor for each object orientation increases the computational cost and may introduce a large number of of potential false-positive detections. the main challenge for linking up 2D and 3D domains lies in the way the objects are represented [35]. representing objects as points greatly simplifies 3D recognition [37]. In this task, we tried to build a 3D that detects centers of the objects and their properties using a key-point detector [39]. In specific we followed the same approach in [37, 9] by using any standard Lidar-based backbone network, for example, PointPillars or VoxelNet, to build an intermediate representation for the input point cloud. this representation is then flattened into Birds-Eye-View map using a standard image-based key-point detector to find object centers [39]. the rest of the 3D bounding box parameters, i.e., 3D Size, orientation, and velocity, are then regressed from the center location vector. according to [37], center-based representation has several advantages: First, it reduces the detector search space since points has no orientation, unlike bounding boxes. Second, center-based representation simplifies the tracking downstream tasks.

Much like 3D perception, Multi-object tracking (MOT) is a crucial component for autonomous driving [18]. Tracking-by-detection paradigm is the dominant approach when it comes to online MOT problems [3, 28]. The main idea is to apply object detector to all frames and extract features independently from each detected object. Then pairwise similarity is computed between objects and a Hungarian algorithm [16] is used to solve the problem. The key idea in this approach is to learn discriminative features for the objects with different identities to reduce the confusion in the matching. One key observation is that feature extraction is done independently and there is no interaction between features during feature extraction. according to [29], independent feature extraction is sub-optimal for discriminative feature learning. this idea is very important for

MOT, given the fact that an object in current frame can be matched to at most one object in the previous frame. so if the similarity between two features increased, then the pairwise similarity between the rest of the objects and any of these two objects should be decreased to avoid confusion for matching. based on that observation, we followed [29] by implementing a Graph Neural Network to do the feature interaction. The idea is to construct a graph with each node being the object feature. Then, each node can update its feature by aggregating features from other nodes during layer propagation step. the object feature is now not isolated and can be adapted with respect to other features.

2. Related Work

3D Object Detection is all about predicting the three dimensional rotated bounding boxes [10, 17, 21, 33, 35, 36]. Most 3D-based methods either use point cloud data directly or require converting these data into 3D grids or voxels instead of generating BEV representations. In [25], point cloud data are converted into voxels containing feature vectors, and then a novel convolution-like voting-based algorithm is used for detection. Vote3Deep [6] leverages feature voting in [25] to efficiently process the sparse 3D point-cloud in equally spaced 3D voxels. VoxelNet [41] uses a PointNet [22] inside each voxel to generate a unified feature representation from which a head with 3D sparse convolutions [12] and 2D convolutions produces detections. SECOND [33] simplifies the VoxelNet and speeds up sparse 3D convolutions. PIXOR [34] project all points onto a 2D feature map with 3D occupancy and point intensity information to remove the expensive 3D convolutions. PointPillars[17] replaces all voxel computation with a pillar representation, a single tall elongated voxel per map location, improving backbone efficiency. MVF [40] and Pillarod [26] combine multiple view features to learn a more effective pillar representation. In this task we followed [37, 9] by focusing on the output representation and employ any 3D encoder to get the representation.

Online Multi-Object Tracking is mostly addressed using Tracking by Detection paradigm. the performance is highly impacted by two factors: object detection quality and discriminative feature learning. After the affinity matrix is computed based on the pairwise similarity of learned discriminative feature, the problem of online MOT could be addressed as as a discrete optimization problem and could be solved using as bipartite matching problem using the Hungarian algorithm [16]. To obtain discriminative feature, prior work mostly focuses on the feature selection. Among different features, it turns out that motion and appearance are the most dis- criminative features. Early work employs hand-crafted features such as spatial distance [20] and Intersection of Union (IoU) [4] as the motion feature, and use color histograms as the appearance feature. Recently, Convolutional Neural Networks have been used to extract appearance feature [1, 7]. Regarding the motion feature, many filter based approaches has been used [3, 28]. Deep learning approaches has been introduced for motion feature in [1]. we followed the same idea in [30, 29] by exploring both the appearance and motion feature with focus on the 3D space.

Graph Neural Networks is used as way to improve discriminative feature learning for MOT after showing promising performance in many fields[13, 24, 15, 2, 19, 38]. GNNs was first proposed by [11] to directly process graph-structured data using neural networks. The major component of the GNNs is the node feature aggregation technique, with which node can update its feature by interacting with other nodes. With this technique, significant success has been achieved in many fields using GNNs such as semantic segmentation[5], action recognition [23], single object tracking [8], person re-identification [31], point cloud classification and segmentation [27]. The majority of the work deals with object features as independent and isolated from other features. by leveraging node aggregation technique of the GNNs, object features could be iteratively evolved so that the feature of different objects can be more discriminative. the idea of feature interaction was first introduced in [14] to encode context information for object detection in the spatial domain, Although a temporal relation network is proposed in the follow-up work [32], the feature of a tracked object is only aggregating from its past trajectory and no interaction with other object features exist. in This task we followed [37] by implementing a generic feature interaction framework that can model any kind of interaction in both spatial and temporal domains

3. Dataset

4. Network Architecture

During the early stages of this task, we tried to use Votenet [**?**] as 3D object detector. Despite the promising results of this architecture on Indoor dataset, it performs very poorly on outdoor datasets like KITTI and Waymo. More information about these experiments are stated Appendix **??**

4.1. Detection

4.2. Tracking

5. Experiments

5.1. Main Results

5.2. Ablation Studies

References

[1] Erkan Baser, Venkateshwaran Balasubramanian, Prarthana Bhattacharyya, and Krzysztof Czarnecki. Fantrack: 3d multi-object tracking with feature association network. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1426–1433. IEEE, 2019.

[2] Rianne van den Berg, Thomas N Kipf, and Max Welling. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263*, 2017.

[3] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)*, pages 3464–3468. IEEE, 2016.

[4] Erik Bochinski, Volker Eiselein, and Thomas Sikora. High-speed tracking-by-detection without using image information. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6. IEEE, 2017.

[5] Yunpeng Chen, Marcus Rohrbach, Zhicheng Yan, Yan Shuicheng, Jiashi Feng, and Yannis Kalantidis. Graph-based global reasoning networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 433–442, 2019.

[6] Martin Engelcke, Dushyant Rao, Dominic Zeng Wang, Chi Hay Tong, and Ingmar Posner. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1355–1361. IEEE, 2017.

[7] Davi Frossard and Raquel Urtasun. End-to-end learning of multi-sensor 3d tracking by detection. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 635–642. IEEE, 2018.

[8] Junyu Gao, Tianzhu Zhang, and Changsheng Xu. Graph convolutional tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4649–4659, 2019.

[9] Runzhou Ge, Zhuangzhuang Ding, Yihan Hu, Yu Wang, Sijia Chen, Li Huang, and Yuan Li. Afdet: Anchor free one stage 3d object detection. *arXiv preprint arXiv:2006.12671*, 2020.

[10] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012.

[11] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734. IEEE, 2005.

[12] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9224–9232, 2018.

[13] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *arXiv preprint arXiv:1706.02216*, 2017.

[14] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3588–3597, 2018.

[15] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[16] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[17] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019.

[18] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 3569–3577, 2018.

[19] Federico Monti, Michael M Bronstein, and Xavier Bresson. Geometric matrix completion with recurrent multi-graph neural networks. *arXiv preprint arXiv:1704.06803*, 2017.

[20] Hamed Pirsiavash, Deva Ramanan, and Charless C Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *CVPR 2011*, pages 1201–1208. IEEE, 2011.

[21] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 918–927, 2018.

[22] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[23] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Skeleton-based action recognition with directed graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7912–7921, 2019.

[24] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[25] Dominic Zeng Wang and Ingmar Posner. Voting for voting in online point cloud object detection. In *Robotics: Science and Systems*, volume 1, pages 10–15607. Rome, Italy, 2015.

[26] Yue Wang, Alireza Fathi, Abhijit Kundu, David Ross, Caroline Pantofaru, Tom Funkhouser, and Justin Solomon.

Pillar-based object detection for autonomous driving. *arXiv preprint arXiv:2007.10323*, 2020.

[27] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019.

[28] Xinshuo Weng and Kris Kitani. A baseline for 3d multi-object tracking. *arXiv preprint arXiv:1907.03961*, 2019.

[29] Xinshuo Weng, Yongxin Wang, Yunze Man, and Kris M Kitani. Gnn3dmot: Graph neural network for 3d multi-object tracking with 2d-3d multi-feature learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6499–6508, 2020.

[30] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE, 2017.

[31] Jinlin Wu, Yang Yang, Hao Liu, Shengcai Liao, Zhen Lei, and Stan Z Li. Unsupervised graph association for person re-identification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8321–8330, 2019.

[32] Jiarui Xu, Yue Cao, Zheng Zhang, and Han Hu. Spatial-temporal relation networks for multi-object tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3988–3998, 2019.

[33] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018.

[34] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7652–7660, 2018.

[35] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11040–11048, 2020.

[36] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Std: Sparse-to-dense 3d object detector for point cloud. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1951–1960, 2019.

[37] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3d object detection and tracking. *arXiv preprint arXiv:2006.11275*, 2020.

[38] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 974–983, 2018.

[39] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019.

[40] Yin Zhou, Pei Sun, Yu Zhang, Dragomir Anguelov, Jiyang Gao, Tom Ouyang, James Guo, Jiquan Ngiam, and Vijay Vasudevan. End-to-end multi-view fusion for 3d object detection in lidar point clouds. In *Conference on Robot Learning*, pages 923–932. PMLR, 2020.

[41] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018.

Appendices

A. Votenet for Lidar Outdoor Scenes

A.1 Data preparation and Training

VoteNet consists of two modules: the proposal module which consumes the raw point cloud and produces the virtual voting points; and the object proposal and classification module that operates on the voting points to propose and classify objects. Up to now, VoteNet has only been tuned and tested on RGB-D data (ScanNet and SUN RGB-D). in order to adapt the model for Waymo, we implemented a custom input pipeline to optimally preprocess and feed the Waymo point cloud data into the VoteNet network. Following the same train/validation/test split from Waymo paper [**?**], and following the same training procedure mentioned in [**?**], For each scene we randomly subsample 20k points. The scenes are furthermore augmented on-the-fly with random flips in horizontal plane, random uniform rotations along up-axis in $\pm 30^{\circ}$ range, and random uniform scaling of $\pm 10\%$. In addition to three Euclidean coordinates for each point, we also include up to two additional features: the provided laser reflection intensities, and a height estimate for each point. The latter is estimated as a 1% percentile of all point positions along the up-axis. We perform training on the dynamic classes of the Waymo dataset: car, pedestrian, and cyclist. We choose one size template per class, and 12 bins for the heading angle. We train with a batch size of 16 and incorporate a scheduled learning policy with a starting LR of 0.001 and LR-decays by 0.1 at 60, 90, and 120 epochs, as well as an exponential BN momentum decay. We use same loss function as in SUN RGB-D case.

Points clouds from outdoor LiDAR scans are quite different from point clouds from RGB-D indoor scans. For example, the typical scales of the Waymo scenes are significantly larger than those of indoor scans, with depth fields reaching beyond 70 meters along forward-axis. the LiDAR point cloud has strongly varying density and is generally more sparse than the point clouds produced from the RGB-D imagery. Point features extracted from sparse regions may generalize poorly to dense regions, and vice versa

to further adapt VoteNet to the distinct characteristics of the outdoor LiDAR scenes. we adapt the receptive field radii and increase the number of clusters for feature aggregation. to capture fine details of point cloud and, at the same time, mitigate the corruption of local patterns due to sampling deficiency in sparse cloud regions, we enhance the set-abstraction modules of the backbone network with multi-scale grouping (MSG) layers [**?**], which concatenate

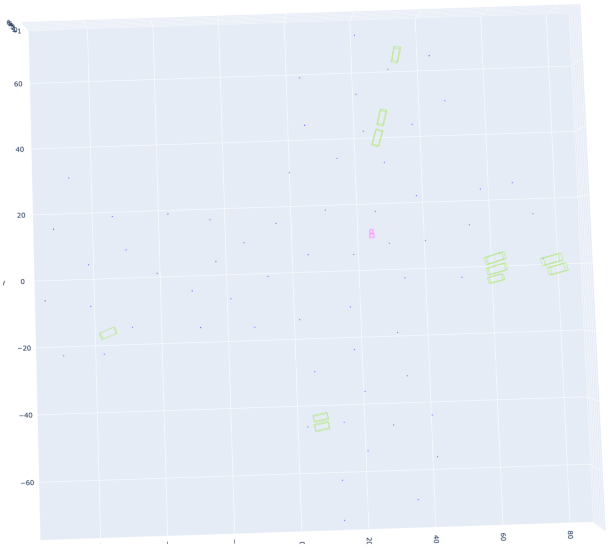

Figure 1. Top view of final 3D space. Blue Points are the final proposal points from VoteNet. Green and Magenta boxes are Ground truth bounding boxes. During training, the final proposal are all lie outside the ground truth bounding boxes. Hence, the voting module cannot learn the correct modules.

features at different scales before feeding them into the feature aggregation layer.

Among all the experiments to find the best backbone network parameters, we ended up following the configuration in [**?**] by configuring four MSG-based SA layers used to subsample the inputs to 4096, 1024, 256 and 64 points, and two FP layers to upsample the points back to 1024 points, each with additional 512 deep features. Tables A.1 and A.1 illustrates the difference between the two backbones.

A.2 Results on 360° Point cloud

When working the 360 point cloud, the Average Precision results from different VoteNet architectures were very poor. after training for nearly 120 epochs, mAP over the test set was nearly 0.15 only. the problem lies in the different points densities between outdoor LiDAR point clouds and indoor RGB-D point cloud. there are a lot of objects which have only 2 10 points per bounding box. after the initial downsampling from 120k to 20k points for processing the further downsampling done by Set abstraction modules, the final proposal points are actually lie outside most of the ground truth bounding boxes as shown in Figure 1. Thus, the network was not able to learn the correct voting. nearly all of the detected objects in this experiment were objects near to the sensor with ¿= 1000 points, all objects with fewer points were missed.

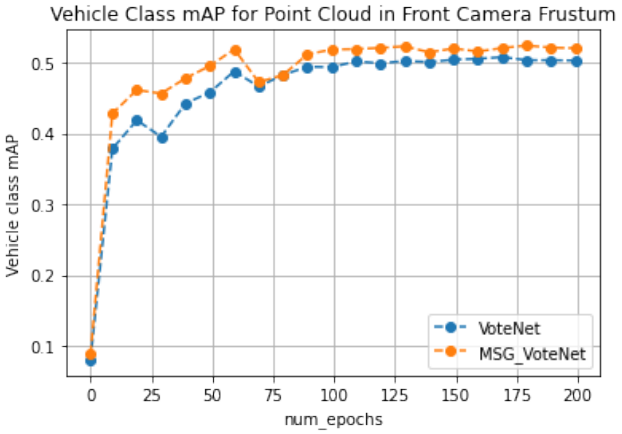

Figure 2. Example of caption. It is set in Roman so that mathematics (always set in Roman: $B \sin A = A \sin B$) may be included without an ugly clash.

A.3 Results on Front Camera Frustum Point Cloud

To verify the results of the previous section, we did an extensive exploration in the literature about the experiments involves raw point processing approaches [**?**, **?**, **?**, **?**, **?**] for outdoor datasets [**?**, **?**, **?**, **?**]. interesting work in [**?**] in forground background segmentation caught our attention. as a first step, we reproduced the work in [**?**] on KITTI dataset [**?**]. Then, Before reproducing the same experiment on Waymo cite, we implemented a specific data processing pipeline to make Waymo point cloud identical to KITTI. As KITTI includes annotations only for the objects in teh front camera frustum. we first project the 360-degree LiDAR data onto the image plane and extract the point cloud that lie inside the result- ing frustum. This step reduces the average number on points per scene from ∼170k to 16,384. For each scene we randomly subsample 16,384 points, and for scenes with fewer points, we randomly copy the points up to a total of 16,384.

reducing the size of the point cloud make a tremendous boost in the (AP) results of VoteNet when we re-executed the experimetents mentioned in **??** on this data setup. due to the small initial input size, the ratio between points inside bounding boxes to the points outside bounding boxes are very large when compared in case of using the 360 point cloud. many of the final proposal points are found inside the ground truth bounding boxes. Thus the voting module was able to learn the correct voting. Figure 2 shows the final results of running two VoteNet architectures on the reduced Waymo point clouds.

B. Appendix Two Test

Minkowski Vs Spconv vs Sparse Tensor library

Module (Input)	Output dimensions	clusters K	Receptive field radius	MLP layer dimensions
SA1 (PC)	(4096, 3 + 96)	2048	0.2	64/64/128
SA2 (SA1)	(1024, 3 + 256)	1024	0.4	128/128/256
SA3 (SA2)	(512, 3 + 512)	512	0.8	128/128/256
SA4 (SA3)	(64, 3 + 512)	256	1.2	128/128/256
FC1 (SA3, SA4)	(512, 3 + 512)	-	-	512/512
FC2 (SA2, SA3)	(1024, 3 + 512)	-	-	512/512

Table 1. Original Architecture of VPN in [**?**]

Module (Input)	Output dimensions	clusters K	MSG Radii	MLP layers
SA1 (PC)	(4096, 3 + 96)	2048	0.1, 0.5	16/16/32, 32/32/64
SA2 (SA1)	(1024, 3 + 256)	1024	0.1, 0.5	64/64/128, 64/96/128
SA3 (SA2)	(512, 3 + 512)	512	0.1, 0.5	128/196/256, 128/196/256
SA4 (SA3)	(64, 3 + 512)	256	0.1, 0.5	256/256/512, 256/256/512
FC1 (SA3, SA4)	(512, 3 + 512)	-	-	512/512
FC2 (SA2, SA3)	(1024, 3 + 512)	-	-	512/512

Table 2. Enhanced architecture of VPN with two MSG radii of receptive fields $r_{1,2}$ (meters), and MLP parameters for each MSG group implemented [**?**]