Relationship Between Semantic Segmentation Model and Additional Features for 3D Point Clouds Obtained from on-Vehicle LIDAR

Hisato Hashimoto and Shuichi Enokida
Graduate School of Computer Science and Systems Engineering, Kyushu Institute of Technology, 680-4 Kawazu, Iizuka-shi, Fukuoka, 820-8502, Japan

Keywords: Semantic Segmentation, Point Cloud, Deep Learning.

Abstract: The study delves into semantic segmentation's role in recognizing regions within data, with a focus on images and 3D point clouds. While images from wide-angle cameras are prevalent, they falter in challenging environments like low light. In such cases, LIDAR (Laser Imaging Detection and Ranging), despite its lower resolution, excels. The combination of LIDAR and semantic segmentation proves effective for outdoor environment understanding. However, highly accurate models often demand substantial parameters, leading to computational challenges. Techniques like knowledge distillation and pruning offer solutions, though with possible accuracy trade-offs. This research introduces a strategy to merge feature descriptors, such as reflectance intensity and histograms, into the semantic segmentation model. This process balances accuracy and computational efficiency. The findings suggest that incorporating feature descriptors suits smaller models, while larger models can benefit from optimizing computation and utilizing feature descriptors for recognition tasks. Ultimately, this research contributes to the evolution of resource-efficient semantic segmentation models for autonomous driving and similar fields.

1 INTRODUCTION

In recent years, recognition technology for the operating environment of autonomous transportation devices, such as self-driving vehicles, has been gaining attention worldwide. Various technical approaches are continuously developed, and among them, semantic segmentation is one effective method. Semantic segmentation enables region recognition by assigning attributes to individual elements of data, and it has gained significant attention, particularly with the advancement of deep learning models. Data used for semantic segmentation processing include images and 3D point clouds. Generally, wide-angle cameras are used for capturing images in vehicles, while LIDAR is employed for obtaining 3D point clouds in broad scenes like traffic environments. Cameras have high resolutions and are commonly used sensors not only in the field of autonomous driving but also in a wide range of applications such as robotics and household appliances. However, cameras can become unstable in environments with wide dynamic ranges, like urban areas at night. On the other hand, while LIDAR has the disadvantage of lower resolution compared to cameras, it can measure without being affected by low-light conditions or external light interference. As shown in Figure 1, in captured camera images, dark areas can result in unclear distinctions, making discrimination difficult in low-light environments. In contrast, LIDAR-derived point clouds can capture the point clouds of multiple individuals even in such low-illumination scenarios. This is because LIDAR is an active sensor while the camera is a passive sensor. Efforts towards achieving autonomous platooning without the need for special road infrastructure(Toshiki Isogai, 2016) also emphasize the ad-

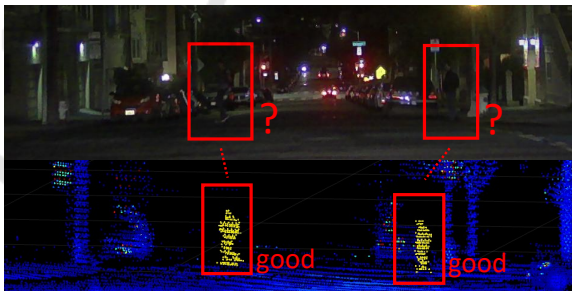

Figure 1: Robust measurement performance of LIDAR in low-light environments. While identification is challenging with camera (above), LIDAR (below) accurately captures the human form(Created from (Xiao et al., 2021)'s data).

Hashimoto, H. and Enokida, S.
Relationship Between Semantic Segmentation Model and Additional Features for 3D Point Clouds Obtained from on-Vehicle LIDAR.
DOI: 10.5220/0012374400003660

In *Proceedings of the 19th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2024) - Volume 2: VISAPP*, pages 547-553
ISBN: 978-989-758-679-8; ISSN: 2184-4321

vantages of the active approach, and LIDAR is being adopted for this purpose. This combination of the robustness to illumination changes offered by LIDAR and the ability to comprehend multiple classes of regions at once through semantic segmentation proves to be an effective approach for outdoor environment recognition. Therefore, in this study, we focus on LIDAR and target semantic segmentation for 3D point clouds acquired by LIDAR.

2 RELATED WORKS

2.1 Image Semantic Segmentation

Semantic segmentation plays a crucial role in tasks involving region recognition within data. While traditional image classification mainly involved abstraction of numerous pixels into higher-dimensional representations, semantic segmentation requires the output of information for each pixel, demanding the restoration of the abstracted information back to the original resolution. Early semantic segmentation methods(Shi and Malik, 2000; Felzenszwalb and Huttenlocher, 2004) represented images as graphs and utilized graph-cut algorithms to perform segmentation based on the relationships between neighboring pixels. Subsequently, with the rapid development of deep learning, the Fully Convolutional Network (FCN)(Long et al., 2015) was introduced, which marked the beginning of semantic segmentation methods using neural network architectures. FCNs, primarily used for semantic segmentation, forgo fully connected layers and instead rely on typical convolutional and pooling layers, thereby preserving position information across the entire input image and enabling image segmentation. Furthermore, the U-Net(Ronneberger et al., 2015) architecture introduced an encoder-decoder pair, enabling finer segmentation by aggregating information spatially and then expanding it back to the original space. The DeepLab series(Chen et al., 2017; Chen et al., 2018) further improved segmentation accuracy by combining atrous convolution and fully connected conditional random fields (CRF).

2.2 Point Cloud Semantic Segmentation

The evolution of research on 2D images gradually began to influence the field of semantic segmentation for 3D point clouds. VoxNet(Maturana and Scherer, 2015) was the first model to convert 3D point clouds into cuboids called voxel grids and apply 3D CNNs. However, voxelization can be computationally expensive and fail to fully account for the disorder and sparsity of 3D data. PointNet(Qi et al., 2017) became the first deep learning model that directly processed individual points independently, thus addressing the disorder and sparsity of point clouds. More recently, SqueezeSeg(Wu et al., 2018) was proposed, enabling deep learning-based semantic segmentation on LIDAR-derived point cloud data. SqueezeSeg leverages the advantages of aligned point cloud data by utilizing a combination of feature extraction from image-based CNNs and class classification through fully connected layers. This approach achieves efficient and accurate segmentation. The advancement of deep learning significantly improved accuracy, but models designed for high accuracy often come with a substantial number of parameters, demanding high computational power. Particularly in the realm of semantic classification models, increasing the number of channels or layers in deep learning models is a straightforward way to improve accuracy, but for deployment on edge devices, aiming for high accuracy while keeping the model small becomes crucial.

3 RELATIONSHIP BETWEEN MODEL SIZE AND ACCURACY

Semantic classification models for point clouds include SqueezeSeg, which employs the image classification CNN SqueezeNet(Iandola et al., 2016), and PointNet, which processes each point through the same Multi Layer Perceptron (MLP) and addresses the invariance of point order through max-pooling. Both are capable of achieving semantic classification for 3D point clouds, but CNNs exhibit better performance compared to MLPs, thanks to their ability to exploit local correlations (as demonstrated by AlexNet(Krizhevsky et al., 2012)). Moreover, CNNs are more efficient and lightweight due to fewer parameters than MLPs, which is why CNN-based models have become mainstream in recognition tasks since the introduction of AlexNet. In general, the accuracy of CNNs tends to increase with more channels in each layer, but this comes at the cost of increased computation and processing time. When models become large, it's common to compress them for deployment on edge devices, achieving lightweight and faster processing. Techniques frequently used for model compression include "knowledge distillation(Hinton et al., 2015)", "quantization(Wu et al., 2020)" and "network pruning(Han et al., 2015)" as discussed by Anthony Berthelier et al.(Anthony et al., 2021). Knowledge distillation involves training a student model to minimize the loss between the output

of the pre-compression teacher model and the post-compression student model, transferring knowledge from the teacher to the student. Quantization reduces the memory usage of model parameters (weights, etc.) by representing them with fewer bits, while keeping the network structure unchanged. Network pruning involves removing nodes with small weights, achieving model compression while maintaining accuracy. However, these methods can lead to significant accuracy degradation depending on the compression goals. Therefore, in this study, we investigate the relationship between various features and model architecture with the goal of lightweighting the model while maintaining comparable accuracy to the original model. This is achieved by combining a reduced architecture obtained by pruning nodes before training with a method for improving accuracy through feature addition.

4 FEATURES EXTRACTED FROM POINT CLOUDS ACQUIRED FROM ON-VEHICLE LIDAR

For the additional features to the model's input in this study, "reflectance intensity", "reflectance intensity histogram" representing the distribution of reflectance intensities in neighboring point clouds, and "Fast Point Feature Histograms (FPFH)(Rusu et al., 2009)" describing surface shapes through histogramizing the distribution of normals based on the normals of the points of interest and their neighbors are used. While 3D coordinates and reflectance intensities, directly obtained from LIDAR, are used as input features, this study investigates the impact of reflectance intensity histograms and FPFH, which describe features outside the model, through comparative experiments to assess their effectiveness.

4.1 Reflectance Intensity

Reflectance intensity is a type of measurement data obtained from LIDAR, calculated from the attenuation rate of the laser light emitted from the LIDAR's light emitter, reflected from the target object, and received by the LIDAR's light receiver. As shown in Figure 2, the reflectance intensity for artificial surfaces mainly composed of black asphalt has a mode value of 106, indicating a narrow distribution. On the other hand, as shown in Figure 3, the reflectance intensity for natural surfaces mainly composed of green grass has a mode value of 1156, stronger than asphalt,

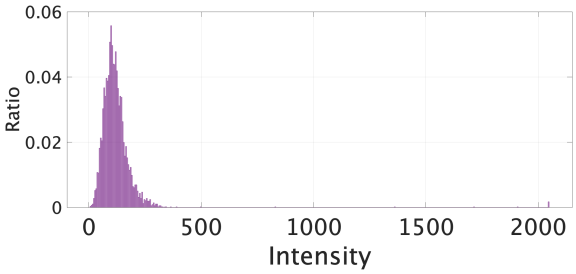

Figure 2: Distribution of reflectance intensity(Asphalt: black artificial object).

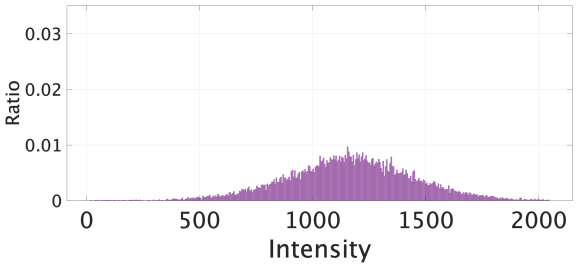

Figure 3: Distribution of reflectance intensity(Grass: green natural object).

with a wider distribution. Therefore, reflectance intensity serves as a feature that can reflect differences in color and between artificial and natural objects.

4.2 Reflectance Intensity Histogram

The reflection intensity histogram is a novel feature defined in this study, which involves histogramizing the occurrence distribution of reflection intensities within the neighborhood points of an interest point using weighted summation (as depicted in Figure 4). When directly summing the occurrence distribution within the neighborhood points, the correlation with the interest point tends to weaken, causing excessive blurring of information. Therefore, in this study, the histogramization is performed while varying the weights based on the distance d_i from the interest point, thereby maintaining the correlation with the interest point. The weight w_i for the i-th point p_i within the neighborhood points is defined using Equation (1), where σ is defined by Equation (2).

$$w_i = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{{d_i}^2}{2\sigma^2}\right) \quad (1)$$

$$\sigma = ar \quad (2)$$

(a : Attenuation Rate, r : Range of Point Search)
This feature's dimensionality corresponds to the number of bins used for partitioning. In this paper, the number of bins was set to 31, with values a and r set to 1.8 and 0.5, respectively.

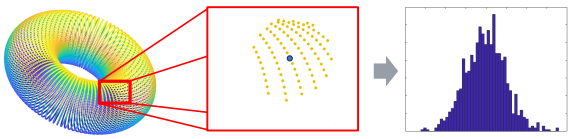

Figure 4: Creating a histogram of reflection intensity, after selecting points within a specified radius from the focal point and creating a local point cloud, the reflectance intensity of points contained in this local point cloud is the subject of the histogram.

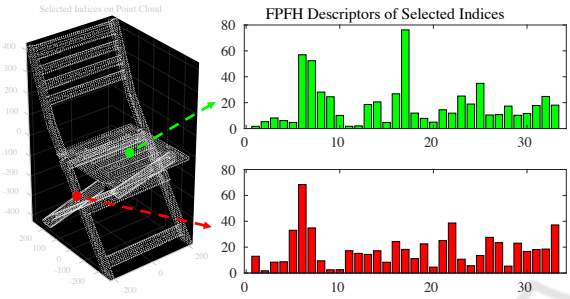

Figure 5: Feature Representation of FPFH. Green point on the monotonous seat of the chair, red point on the complex legs of the chair.

4.3 Fast Point Feature Histograms(FPFH)

Fast Point Feature Histograms (FPFH) is a type of local feature descriptor computed from 3D point cloud data, aiming to achieve effective feature representation (33 dimensions) while maintaining compactness. FPFH describes the surface shape by histogramizing the distribution relationship of normals between an interest point and its neighboring points based on their normals. The foundation of FPFH, Point Feature Histograms (PFH)(Rusu et al., 2008b)(Rusu et al., 2008a), captures geometric features around keypoints by analyzing the normals' directions of the keypoints' vicinity. PFH creates pairs among all neighboring points to build histograms, ensuring accurate results, but it suffers from a significant computational cost. In a point cloud with n points, the computational complexity of PFH for a neighborhood size k is $O(nK^2)$. To mitigate this computational burden, FPFH uses only the links between keypoints and their neighboring points. By reducing these links, FPFH reduces the computational complexity to $O(nK)$. Figure 5 illustrates an example of computed FPFH. Differences in histograms can be observed between the seat (red points) and legs (green points) of the chair.

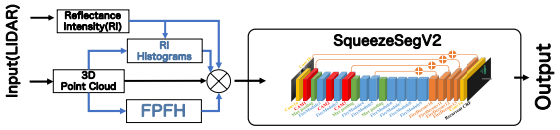

Figure 6: Input structure of features. Before inputting into SqueezeSegV2, combine intensity, intensity histogram and FPFH with 3D coordinates(xyz).

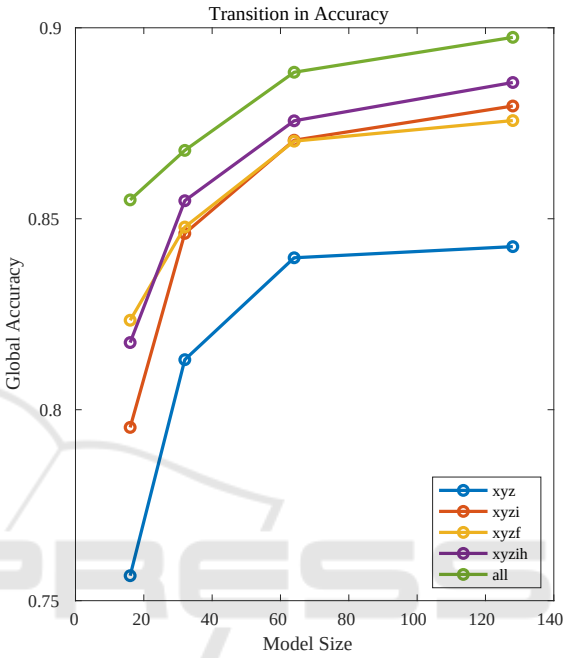

Figure 7: Accuracy transitions with changes in model size(Plotting at four different model size : 16, 32, 64, 128), blue: xyz only, red: with intensity, yellow: with FPFH, purple: intensity histogram, green: with all features.

5 EXPERIMENTS AND DISCUSSIONS COMPARING THE EFFECTIVENESS OF VARIOUS FEATURES

In this study, a comparative experiment is conducted using PandaSet(Xiao et al., 2021), which consists of 3D point cloud data collected from onboard LIDAR sensors. The SqueezeSegV2 (Wu et al., 2019) architecture is employed to assess the effectiveness of various features. The experiment is structured as shown in Figure 6. SqueezeSegV2 employs multiple layers of encoder modules to aggregate features, and the base channel numbers (16, 32, 64, 128) that constitute each layer are considered as the model sizes. The number of layers is kept fixed. Five input patterns are investigated: [xyz: 3D coordinates only, xyzi: with added intensity, xyzih: with added intensity his-

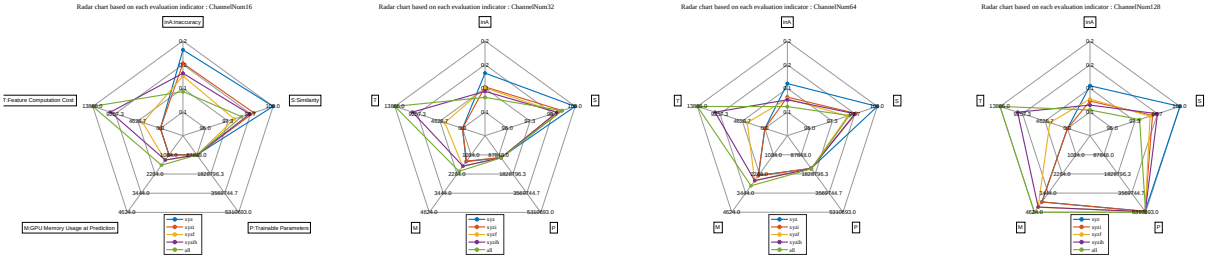

Figure 8: Radar chart based on each evaluation indicator(Model size from left to right: 16, 32, 64, 128), blue: xyz only, red: with intensity, yellow: with FPFH, purple: intensity histogram, green: with all features.

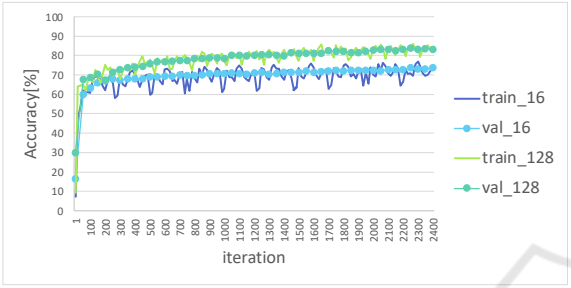

Figure 9: [xyz-Only] Accuracy Transition During Training for Maximum(128) and Minimum(16) Model Sizes.

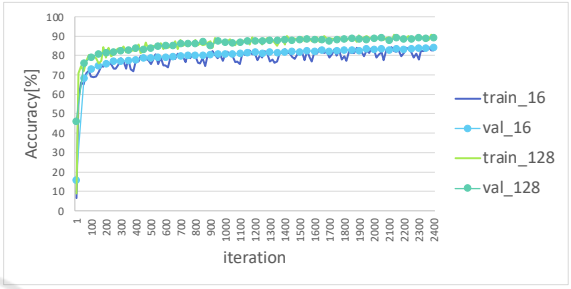

Figure 11: [all-Features] Accuracy Transition During Training for Maximum(128) and Minimum(16) Model Sizes.

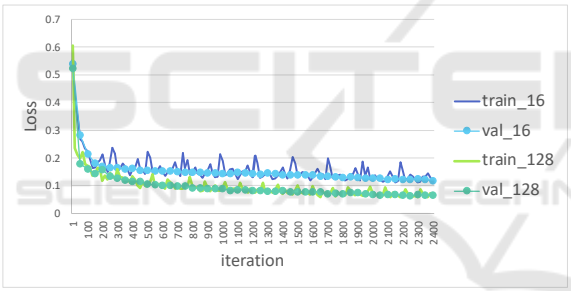

Figure 10: [xyz-Only] Loss Transition During Training for Maximum(128) and Minimum(16) Model Sizes.

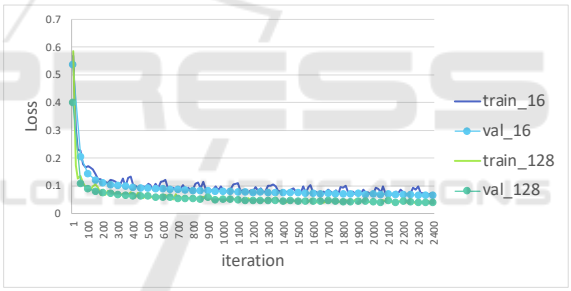

Figure 12: [all-Features] Loss Transition During Training for Maximum(128) and Minimum(16) Model Sizes.

togram, xyzf: with added FPFH, all: with all features added]. These input patterns are evaluated using four model size variations (a total of 20 patterns) in the range of model size variations mentioned above. The evaluation is performed through three rounds of training and testing, and the results are averaged for analysis.

Figure 7 shows the transition in accuracy as the model size is changed. From Figure 7, it is evident that with model sizes of 32 for 'xyzi', 'xyzih', and 'xyzf', and 16 for all, an accuracy of 0.8441 or higher is achieved, comparable to the model size of 128 for xyz. Additionally, it was confirmed that the convergence of accuracy and loss becomes faster by adding features, as observed in the transition results of accuracy shown in Figure 9 and Figure 11, and the transition results of loss shown in Figure 10 and Figure 12 during training. Moreover, the increase in GPU memory due to feature addition is minimal compared to the increase observed during model size expansion. Therefore, it can be concluded that feature addition is an effective approach for devices with stringent GPU memory constraints, such as edge computing environments.

Additionally, to assess the impact of various features on the model, five metrics are utilized: "Inaccuracy: inA", "Similarity: S", "Trainable Parameters: P", "GPU Memory Usage at Prediction: M", and "Feature Computation Cost: T". These metrics are used to create a radar chart for a comprehensive evaluation. In this study, similarity refers to quantifying the influence of each feature by vectorizing class-wise accuracy when using various features together. This is achieved by calculating the cosine similarity among the vectors. The relative values are calculated by comparing them to the results obtained with only 3D co-

ordinates for the same model size. When adding the additional feature F, the accuracies of all 13 classes are treated as a vector $\boldsymbol{ClassAcc}_F$ with dimensions 1×13. The cosine similarity S is calculated by finding the average cosine similarity between the $\boldsymbol{ClassAcc}_F$ vectors obtained for each feature addition. This cosine similarity serves as a measure of the impact of each feature on the class accuracies. Smaller values indicate that a feature has a different impact on class accuracies compared to other features. Feature computation cost is evaluated using a relative scale based on the time it takes to calculate the feature. The computation time strongly depends on the CPU performance, so a relative scale is defined using the computational cost of calculating the Euclidean distance as 1 and the cost of calculating the non-needed intensity as 0. It can be observed that there is a change in similarity depending on the size of the model. Furthermore, with an increase in model size, there is an increase in GPU memory usage and trainable parameter count. However, the increase in these values due to the addition of features is marginal. While adding all features yields positive impacts on inaccuracy, the computational cost of feature computation becomes economically impractical. On the other hand, while feature addition doesn't significantly affect GPU memory usage during inference, it does involve computation costs. Therefore, selecting appropriate features based on the available CPU and GPU resources is crucial when embedding the model into edge computing devices. Consequently, adding feature-descriptive features is advantageous for smaller models, while for larger models, it's more beneficial to leave the feature description to the model itself to minimize feature computation costs.

6 CONCLUSIONS

In this paper, we propose a method to incorporate additional features into the input of a semantic classification model. While evaluating the effectiveness of these additional features, we aim to achieve both model lightweighting and accuracy preservation. Moving forward, we will focus on observing changes in learning efficiency through feature integration, and strive to adapt the model to be more compact while maintaining high accuracy. Additionally, we will explore comprehensive evaluation methods that encompass various performance metrics.

REFERENCES

Anthony, B., Thierry, C., Stefan, D., Christophe, G., and Christophe, B. (2021). *Deep Model Compression and Architecture Optimization for Embedded Systems: A Survey*. Journal of Signal Processing Systems, vol.93, pp.863-878.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2017). *Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs*. IEEE transactions on pattern analysis and machine intelligence, vol.40, no.4, pp.834-848.

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., and Adam, H. (2018). *Encoder-decoder with atrous separable convolution for semantic image segmentation*. Proceedings of the European conference on computer vision (ECCV), pp.801-818.

Felzenszwalb, P. F. and Huttenlocher, D. P. (2004). *Efficient graph-based image segmentation*. International journal of computer vision, vol.59, pp.167-181.

Han, S., Mao, H., and Dally, W. J. (2015). *Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding*. arXiv preprint arXiv:1510.00149.

Hinton, G., Vinyals, O., and Dean, J. (2015). *Distilling the knowledge in a neural network*. arXiv preprint arXiv:1503.02531.

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., and Keutzer, K. (2016). *SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and¡ 0.5 MB model size*. arXiv preprint arXiv:1602.07360.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). *Imagenet classification with deep convolutional neural networks*. Advances in neural information processing systems, vol.25.

Long, J., Shelhamer, E., and Darrell, T. (2015). *Fully convolutional networks for semantic segmentation*. Proceedings of the IEEE conference on computer vision and pattern recognition, pp.3431-3440.

Maturana, D. and Scherer, S. (2015). *Voxnet: A 3d convolutional neural network for real-time object recognition*. 2015 IEEE/RSJ international conference on intelligent robots and systems (IROS), pp.922-928.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017). *Pointnet: Deep learning on point sets for 3d classification and segmentation*. Proceedings of the IEEE conference on computer vision and pattern recognition, pp.652-660.

Ronneberger, O., Fischer, P., and Brox, T. (2015). *U-net: Convolutional networks for biomedical image segmentation*. Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18, pp.234-241.

Rusu, R. B., Blodow, N., and Beetz, M. (2009). *Fast point feature histograms (FPFH) for 3D registration*. 2009 IEEE international conference on robotics and automation, pp.3122-3217.

Rusu, R. B., Blodow, N., Marton, Z. C., and Beetz, M. (2008a). *Aligning point cloud views using persistent feature histograms*. 2008 IEEE/RSJ international conference on intelligent robots and systems, pp.3384-3391.

Rusu, R. B., Marton, Z. C., Blodow, N., and Beetz, M. (2008b). *Learning informative point classes for the acquisition of object model maps*. 2008 10th International Conference on Control, Automation, Robotics and Vision, pp.643-650.

Shi, J. and Malik, J. (2000). *Normalized cuts and image segmentation*. IEEE Transactions on pattern analysis and machine intelligence, Vol.22 no.8, pp.888-905.

Toshiki Isogai, Mitsuyasu Matsuura, T. K. (2016). *Emvironment Recognition Technology for Trucks platooning*. DENSO TECHINICAL REVIEW, vol.21, pp.71-76.

Wu, B., Wan, A., Yue, X., and Keutzer, K. (2018). *Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud*. 2018 IEEE international conference on robotics and automation (ICRA), pp.1887-1893.

Wu, B., Zhou, X., Zhao, S., Yue, X., and Keutzer, K. (2019). *Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud*. 2019 international conference on robotics and automation (ICRA), pp.4376-4382.

Wu, H., Judd, P., Zhang, X., Isaev, M., and Micikevicius, P. (2020). *Integer quantization for deep learning inference: Principles and empirical evaluation*. arXiv preprint arXiv:2004.09602.

Xiao, P., Shao, Z., Hao, S., Zhang, Z., Chai, X., Jiao, J., Li, Z., Wu, J., Sun, K., Jiang, K., et al. (2021). *Pandaset: Advanced sensor suite dataset for autonomous driving*. 2021 IEEE International Intelligent Transportation Systems Conference (ITSC), pp.3095-3101.