Turkish Journal of Science & Technology
20(1), 327-337, 2025

Research Paper
https://doi.org/10.55525/tjst.1637713

Exploring the Impact of Model Capacity and Parameter Tuning on 3D Semantic Segmentation

Furkan KARAMAN[1*], Fatma GUMUS[2]
[1] Department of Computer Engineering, ATASAREN, National Defence University, İstanbul, Türkiye
[2] Department of Computer Engineering, National Defence University, İstanbul, Türkiye
[*1] furkan09karaman@gmail.com, [2] fatma.gumus@msu.edu.tr

(Geliş/Received: 11/02/2025; Kabul/Accepted: 18/03/2025)

Abstract: 3D semantic segmentation, the process of assigning semantic labels to every point in a 3D space, is critical for numerous applications, including autonomous driving, robotics, medical imaging, and urban mapping. Despite significant progress, challenges such as data imbalance, scalability, and real-time processing constraints persist. This study addresses the real-time processing issue by comparing Tiny, Medium, and Large PointNet-inspired models utilizing the ShapeNetCore dataset. The models incorporate the T-Net module for pose normalization to maintain robustness against geometric transformations. Class-specific segmentation is explored by training separate models for the Airplane, Motorbike, and Car classes, allowing custom optimizations for each class. The Tiny model with 512 sampled points where the batch size is 16 and trained for 40 epochs with a starting learning rate of 1×10^{-3} achieved an average training accuracy of 86.18% and an average validation accuracy of 83.50%, making it optimal for real-time applications due to its fast inference speed and high accuracy.

Key words: 3D semantic segmentation, point cloud processing, scalability, real-time processing.

3B Semantik Bölütleme Performansı Üzerinde Model Kapasitesi ve Parametre Ayarının Etkisinin Araştırılması

Öz: 3B semantik bölütleme, üç boyutlu uzaydaki her noktaya anlamsal etiketler atama sürecidir ve otonom sürüş, robotik, tıbbi görüntüleme ve kentsel haritalama dahil olmak üzere çok sayıda uygulama için kritik öneme sahiptir. Önemli ilerlemeye rağmen, veri dengesizliği, ölçeklenebilirlik ve gerçek zamanlı işleme kısıtlamaları gibi zorluklar devam etmektedir. Bu çalışma, ShapeNetCore veri setini kullanan Tiny, Medium ve Large olarak PointNet'ten esinlenen modelleri karşılaştırmak suretiyle gerçek zamanlı işleme sorununu ele almaktadır. Modeller, geometrik dönüşümlere karşı gürbüzlüğü korumak üzere poz normalizasyonu için T-Net modülünü içerir. Uçak, Motosiklet ve Araba sınıfları için ayrı modeller eğitilerek sınıf-özel segmentasyon çalışılmış ve her sınıf için özel optimizasyon değerlendirilmiştir. Küme büyüklüğünün 16 olduğu ve 1×10^{-3} başlangıç öğrenme oranıyla 40 epok boyunca eğitilen 512 örneklenmiş noktaya sahip Tiny modeli, %86,18 ortalama eğitim doğruluğu ve %83,50 ortalama doğrulama doğruluğu elde etti ve test hızı ve yüksek doğruluğu nedeniyle gerçek zamanlı uygulamalar için ideal olduğu değerlendirilmiştir.

Anahtar kelimeler: 3B semantik bölütleme, nokta bulutu işleme, ölçeklenebilirlik, gerçek zamanlı işleme.

1. Introduction

The rapid advancements in 3D semantic segmentation have significantly improved the ability to analyze and interpret point cloud data. Applications such as autonomous navigation, medical imaging, and remote sensing rely on robust segmentation models to classify and segment objects in complex 3D environments. Despite notable progress, achieving high segmentation accuracy while maintaining computational efficiency remains a critical challenge, particularly for real-time applications. A deep learning model PointNet [1] and its derivatives such as [2] leverage hierarchical feature extraction to capture both local and global geometric properties, enabling precise object classification and segmentation. Voxel-based approaches, on the other hand, convert point clouds into structured 3D grids, enabling the use of convolutional operations [3-5]. Hybrid approaches combine both point cloud and voxel-based methods [6-8].

In contrast to voxel-based and hybrid methods, which introduce quantization artifacts and impose high memory requirements, PointNet operates directly on point clouds, preserving geometric details while maintaining efficiency. This study systematically investigates the influence of architectural and hyperparameter decisions on the segmentation performance of a PointNet inspired architecture. It examines how sampling density affects accuracy, the role of batch size in convergence speed, and the trade-offs between model capacity and segmentation

[*] Corresponding author: furkan09karaman@gmail.com. ORCID Number of authors: [1] 0009-0009-1304-8103, [2] 0000-0001-5191-0037

quality. The paper also emphasizes the importance of real-time applicability, shedding light on a frequently overlooked but crucial factor in 3D segmentation methodologies. This work shares results on class-specific segmentation where a separate segmentation model was trained for each class independently. This approach of segmentation allows custom optimization such that each class can have its architecture, hyperparameters, and loss function. This is also better for imbalanced data; since each model only focuses on one class, it can be better suited for handling rare classes. There are some downsides to this approach, such as if multiple models predict the same pixel, conflict resolution might be required. This paper does not address the approach but presents fine-tuned results for models with different capacities (Tiny, Medium, Large) and three sets of hyper-parameters.

The remainder of this paper is structured as follows: Section 2 provides a comprehensive literature review, summarizing key developments in 3D semantic segmentation, including deep learning-based methodologies, benchmark datasets, and open challenges. Section 3 presents the material and methods. Section 4 presents our experimental results, detailing the dataset, preprocessing steps, and PointNet parameter tuning strategy. Section 5 discusses challenges and open issues in the field, such as data imbalance, scalability, and annotation costs.

2. Literature Review

Point cloud-based methods operate directly on unstructured 3D point clouds, allowing for flexible and efficient data processing. PointNet [1] was a pioneering architecture that employed permutation-invariant networks to process unordered point clouds to perform three major tasks: classification, part segmentation, and semantic segmentation (Figure 1). To prevent sensitivity to point ordering, it uses max pooling. Each point is processed via Multi-Layer Perceptrons (MLPs), and the maximum is computed across all points to learn the global features of the scene. The method's primary advantage is its ability to directly handle raw point cloud data with high speed. PointNet struggled to capture local geometric features, which was addressed by PointNet++ [2]. Instead of using a fixed grid (like CNNs in images), PointNet++ dynamically groups points based on their geometric proximity using a radius threshold. This allows it to adaptively learn geometric details at different scales, preserving local curvature and surface variations. While it excelled in handling complex geometries, it required higher computational costs and processing time. Dynamic Graph CNNs (DGCNN) [9] further expanded on this concept by introducing dynamic graph construction, which updates the neighborhood connections during training. Despite its success, the method required significant computational resources, limiting its scalability for large datasets.

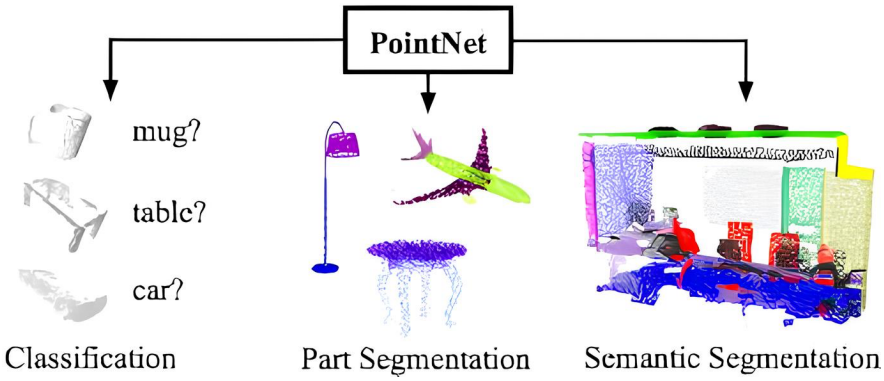


Figure 1. PointNet tasks [1].

Sub-manifold Sparse Convolutional Networks (SSCN) [10] utilized sparse convolutions for more efficient processing of 3D grids. MinkowskiNet [3] extended this concept by using sparse convolutions in 4D spatio-temporal networks. However, voxelization leads to quantization errors, which may degrade segmentation performance at high resolutions. It employs 4D sparse convolutions to optimize time and memory usage for large-scale datasets. Sparse convolutions operate only on active voxels, enhancing memory efficiency.

PolarNet [5] introduced a polar grid representation to improve the processing of LiDAR data, especially when dealing with uneven data distributions. This method demonstrated notable improvements in datasets such as SemanticKITTI [4]. However, it faced challenges related to memory consumption and computational complexity, particularly for large-scale applications. PolarNet processed LiDAR data using polar grid representations. It aimed to understand the global context of the data. The model processed each cell to learn the global features of the scene.

KPConv [11] achieved high accuracy in complex scenes by leveraging deformable convolutions that adapt to irregular geometries. It defines learnable kernel points between points, and filters are automatically shaped based on the density of the data. While it was highly effective in detailed local feature learning, it required substantial computational resources and a complex training process.

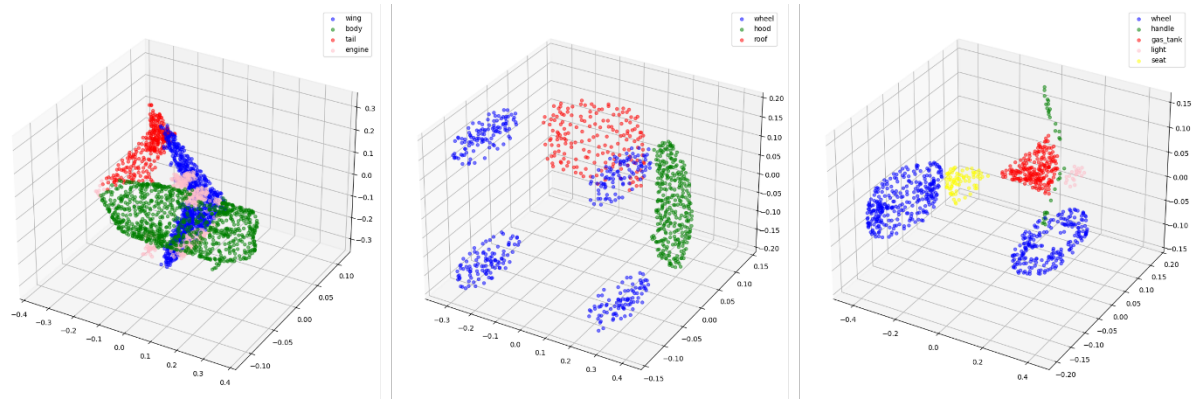
Hybrid methods aim to combine the strengths of point-based and voxel-based techniques. SalsaNext [6] is such a model that integrates point-wise uncertainty estimation with voxel-based representations. It achieved real-time performance for applications like autonomous driving. Using an encoder-decoder architecture, it learned features from LiDAR inputs. The model optimized computations with polar grid representations and enhanced reliability through uncertainty estimation in segmentation predictions. This method is particularly effective for real-time applications, such as environmental sensing in autonomous vehicles. PVCNN [12] is another hybrid approach that integrates the computational efficiency of voxel grids with the flexibility of point-level representations. It offered a balanced approach for real-time applications. SPVConv [7] merged sparse voxel processing with high-resolution point features, resulting in enhanced segmentation accuracy while maintaining computational efficiency. These hybrid models are well-suited for real-world applications that require both precision and high throughput.

SASSNet [8], a semi-supervised learning approach, demonstrated state-of-the-art performance in kidney tumor segmentation. The KiTS19 challenge [13] highlighted the potential of automated 3D segmentation techniques for kidney tumor analysis, showcasing the effectiveness of these methods when annotated data is scarce and expensive to obtain.

The field of 3D segmentation has been extensively reviewed in the literature. Vinodkumar et.al.[14] extensively explored deep learning-based methodologies for 3D object segmentation, detection, and classification. Their work categorized methods based on data modalities, such as LiDAR-based approaches, point cloud techniques, and hybrid models, providing insights into their relative strengths and limitations. Moreover, the study offered an analysis of benchmark datasets like SemanticKITTI and ModelNet, enabling comparative evaluations of various methods. Similarly, He et al. [15] conducted an in-depth survey focusing on three primary segmentation tasks: semantic segmentation, instance segmentation, and part segmentation.

3. Material and Methods

For this study, we utilize ShapeNetCore [16], a subset of the ShapeNet dataset, which is a large-scale, richly annotated repository of 3D shapes. ShapeNetCore includes 55 common object categories with 3D models, providing a reliable benchmark for 3D semantic segmentation. The experiments focused on the three vehicle categories, which are also included as part of the PASCAL 3D+ [17] dataset. The experiments were conducted on the classes: airplane, car, and motorbike, each segmented into specific parts (Table 1). The dataset consists of point cloud representations of 3D models along with manually verified category and alignment annotations (Figure 2). A class-specific segmentation method was used, where each class had its model trained separately. This allows the model to optimize for the unique features of each class.



(a) (b) (c)

Figure 2. Samples from the dataset: a)Airplane b)Car c)Motorbike.

Table 1. Data distribution.

Class	Segmentation parts	Training size	Validation size
Airplane	wing, body, tail, engine	2955	739
Car	wheel, hood, roof	5420	1356
Motorbike	wheel, handle, gas tank, light, seat	187	47

3.1. Preprocessing

One of the key challenges in handling point cloud data is that different 3D scans contain a variable number of points. This variability makes batch processing difficult. To ensure uniformity, we perform the following preprocessing steps:

1) Fixed-size Sampling: We randomly sample a fixed number of points from each point cloud, choosing from $\{256, 512, 1024\}$ to analyze its impact on performance.

2) Normalization: The point clouds are normalized to ensure scale invariance, making the model robust to different object sizes.

3.2. Model variants and hyperparameters

We employ a PointNet-inspired architecture. The proposed model processes unstructured 3D point cloud data to classify each point into predefined segment categories. The architecture consists of three main components: an input transformation block, a hierarchical feature extraction network, and a segmentation head.

Segmentation outputs should remain unchanged under geometric transformations (e.g., translation or scaling). To achieve pose normalization, we apply rigid or affine transformations to input point clouds. This is done using a Spatial Transformer Network (STN) [18], implemented as the T-Net module in PointNet. The T-Net learns a 3×3 transformation matrix using a multi-layer perceptron (MLP), max-pooling, and fully connected layers. This matrix is then applied to the input before feature extraction, ensuring transformation invariance. While global features are sufficient for classification, segmentation requires capturing both local and global contexts. PointNet integrates local point features with global shape descriptors, ensuring fine-grained segmentation. To systematically explore the parameter space, we experiment with the three sets of hyper-parameters in Table 2 and the learning rate scheduled for decay by half every 5 epochs.

Table 2. Hyper-parameter sets.

Parameter set	Number of sampled points	Batch size	Training epochs	Initial learning rate
Set-0	256	8	20	1×10^{-2}
Set-1	512	16	40	1×10^{-3}
Set-2	1024	32	60	1×10^{-4}

We evaluate three different model capacities (Table 3): The large model serves as an upper bound for computational complexity, while the tiny model allows efficient benchmarking with minimal hardware requirements. We ran the experiments on NVIDIA L4 Tensor Core GPU for each hyper-parameter set in Table 2 and the models in Table 3.

Table 3. Number of filters in T-net and segmentation-net convolutions and network size.

Model Variant	Transformation-net	Segmentation-net	Network size (Total parameters / Model Size)
Tiny	16, 32, 256, 128, 64	16, 32, 32, 32, 128, 512	~344K / 1.31 MB
Medium	64, 128, 1024, 512, 256	64, 128, 128, 128, 512, 2048	~7M /28.11 MB
Large	128, 256, 2048, 1024, 512	128, 256, 256, 256, 1024, 4096	~45M /173.22 MB

3.3. The architecture

The segmentation pipeline consists of three main components: input transformation, hierarchical feature extraction, and segmentation output (Figure 3). To ensure pose invariance, an affine transformation was applied to the input point clouds using a Spatial Transformer Network (STN). This module learns a transformation matrix via a multi-layer perceptron (MLP), max-pooling, and fully connected layers. This transformation matrix normalizes point positions, reducing sensitivity to spatial distortions.

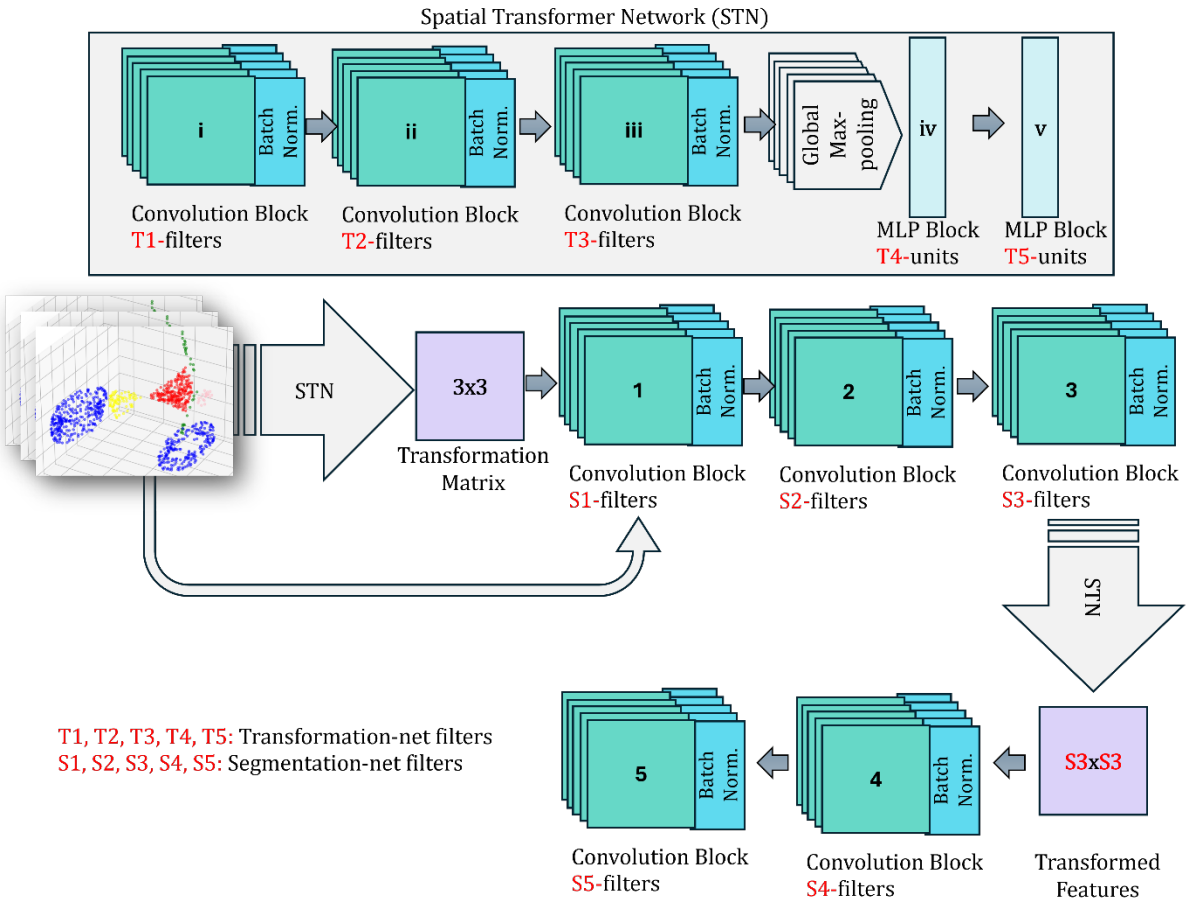


Figure 3. The segmentation pipeline.

The model extracts feature through multiple convolutional layers, progressively capturing local and global geometric structures. The key steps in this pipeline are point-wise feature learning, feature propagation, and global feature aggregation. A series of 1D convolutional layers with ReLU activation map each point to a higher-dimensional feature space. Intermediate features are aggregated across different scales, allowing for both local and global context preservation. Using a max-pooling operation, a global shape descriptor is generated, summarizing the overall object structure while retaining permutation invariance.

The extracted hierarchical features are concatenated and passed through a final segmentation head, which consists of 1D convolutional layers followed by a softmax activation function. This module assigns a class label to each point in the input cloud, completing the segmentation process.

4. Experimental Results

The results in Tables 4-7 highlight the trade-offs between model size, training time, inference speed, and segmentation accuracy. Before class specific insights, a general overview is presented. Tiny models (Table 4) demonstrate the lowest computational overhead, with training times per epoch under 7 seconds and inference times per sample below 2 seconds across all hyperparameter sets. These models provide a fast and lightweight solution while maintaining competitive accuracy.

Table 4. Results for model Tiny.

Class	Hyper-parameters	Training time per epoch (Sec.s)	Inference time per sample (Sec.s)	Training loss	Validation loss	Training accuracy	Validation Accuracy
Airplane	**Set-0**	4.1349	1.7620	2.3877	2.4146	0.8780	0.8698
Airplane	**Set-1**	2.3104	1.7696	8.6765	8.6967	0.8840	0.8740*
Airplane	**Set-2**	2.4414	1.8607	34.0441	34.0239	0.8712	0.8708
Car	**Set-0**	6.8611	1.8447	2.2676	2.3316	0.9157	0.8872
Car	**Set-1**	3.1754	1.6191	8.5696	8.5645	0.9204	0.9153*
Car	**Set-2**	3.7116	1.8042	33.9594	33.7699	0.9095	0.9091
Motorbike	**Set-0**	2.0226	1.8694	2.4935	2.6193	0.8071	0.7603*
Motorbike	**Set-1**	1.2687	1.9001	8.5986	8.7146	0.8039	0.7519
Motorbike	**Set-2**	0.9450	1.9736	33.1477	29.3361	0.7664	0.6769
Average		**2.9857**	**1.8226**	**14.9050**	**14.4968**	**0.8618**	**0.8350**

Medium models (Table 5) increase the parameter count significantly, leading to longer training times (up to 14 s/epoch) and inference times exceeding 2 seconds. In addition to the increased computational cost, the Medium model achieves worse validation accuracy in Set-1 for Airplane (87.40%→86.31%), Car (91.53%→89.14%), and Motorbike (75.19%→71.38%) compared to the Tiny model.

Table 5. Results for model Medium.

Class	Hyper-parameters	Training time per epoch (Sec.s)	Inference time per sample (Sec.s)	Training loss	Validation loss	Training accuracy	Validation Accuracy
Airplane	**Set-0**	1.8816	2.2174	8.5226	8.4514	0.8385	0.8457
Airplane	**Set-1**	3.5018	2.0405	32.9312	32.8586	0.8593	0.8631*
Airplane	**Set-2**	8.5038	2.0583	132.2373	131.8697	0.8130	0.8191
Car	**Set-0**	9.1773	3.1154	15.1606	19.3051	0.8218	0.8126
Car	**Set-1**	8.2329	2.2342	32.8249	34,8191	0.8971	0.8914*
Car	**Set-2**	14.1265	2.1972	131.6679	131.4640	0.8797	0.8840
Motorbike	**Set-0**	3.1759	2.9597	8.7785	23.2788	0.7758	0.6799
Motorbike	**Set-1**	1.6189	2.0418	37.5718	107.0709	0.7741	0.7138*
Motorbike	**Set-2**	1.6044	2.1266	221.6887	9.722.6299	0.7405	0.7093
Average		**6.3708**	**2.3323**	**69.0426**	**1.134.6386**	**0.8222**	**0.8021**

Large models (Table 6) introduce a major computational burden, requiring 12 to 33 s/epoch of training for the large sets of Airplane and Car classes. Validation accuracy drops yet again for most cases, except Set-2 of Airplane. In short, the increased network size and parameter count bring diminishing returns across most of the experiments. As for the Motorbike class, validation accuracy remains inconsistent and sometimes higher than for the Tiny model. This inconsistent behavior is best observed in terms of validation loss.

Table 6. Results for model Large.

Class	Hyper-parameters	Training time per epoch (Sec.s)	Inference time per sample (Sec.s)	Training loss	Validation loss	Training accuracy	Validation Accuracy
Airplane	**Set-0**	12.5527	1.9552	16.6900	16.6852	0.8040	0.8027
Airplane	**Set-1**	13.2459	2.0164	70.2028	73.5573	0.8146	0.8151
Airplane	**Set-2**	19.8034	0.1301	337.5113	322.4445	0.8311	0.8377*
Car	**Set-0**	18.8614	2.2012	16.6593	18.9986	0.8238	0.8058
Car	**Set-1**	21.1859	2.1274	65.7322	70.3808	0.8334	0.8385*
Car	**Set-2**	33.2247	2.3344	275.3770	287.1585	0.8240	0.8240
Motorbike	**Set-0**	4.8082	2.0731	62.1769	9503.0341	0.7364	0.6749*
Motorbike	**Set-1**	2.3329	2.0818	217.9921	675.8121	0.7580	0.6280
Motorbike	**Set-2**	2.7758	3.1757	1080.8776	170735.4062	0.7384	0.6785*
Average		**14.3101**	**2.0106**	**238.1355**	**20189.2753**	**0.7960**	**0.7672**

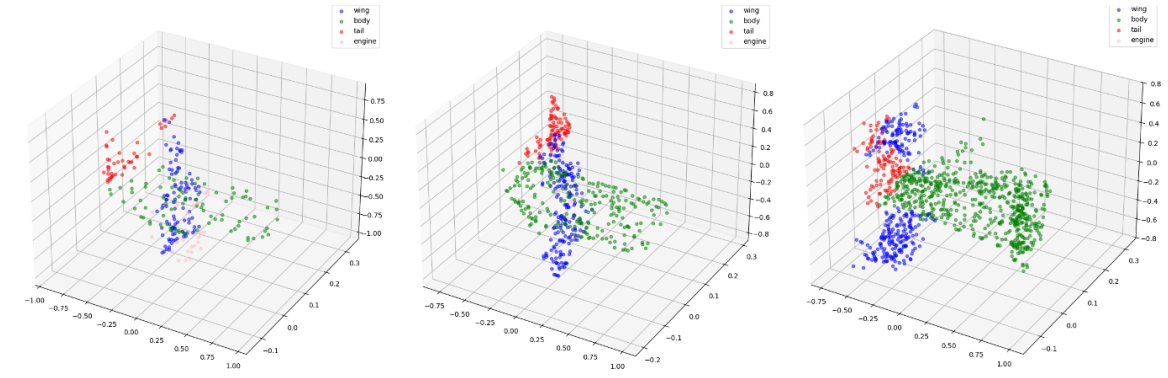
4.1. Segmentation of Airplane

Hyperparameter Set-0 indicates a small point cloud with a high learning rate, Set-1 moderate point cloud with reduced learning rate, and Set-2 largest point cloud with the lowest learning rate. Set-0 has the shortest training time across all model sizes as expected. However, higher training and validation loss values indicate potential convergence issues. Set-1 achieves the best balance between training time and segmentation accuracy. The medium model in Set-1 reaches 86.31% validation accuracy, making it a strong candidate for practical deployment. Set-2 shows stability with lower training loss but at a significantly higher computational cost. The large model in Set-2 reaches 83.77% accuracy but suffers from extreme inference latency (337.51s per sample), making it impractical for real-time scenarios.

The Large Models show a notable increase in computational time and memory usage but do not achieve the highest accuracy when compared to the Tiny and Medium Models. In Set-1, the Large Model has a validation accuracy of 81.51%, which is lower than both the Tiny Model (87.40%) and the Medium Model (86.31%). This suggests that, despite the larger model capacity, it may not be as well-optimized for the task at hand, likely due to overfitting or the challenges of efficiently capturing the relevant features with a larger network.

Figure 4 shows the segmentation results on medium models under each parameter set. The qualitative results indicate that Set-1 achieves a well-balanced trade-off between segmentation accuracy and computational efficiency. While Set-0 exhibits faster training times, its segmentation outputs show slightly less refined boundaries, particularly in complex regions of airplane models. In contrast, Set-2, which uses the highest number of sampled points and training epochs, produces finer segmentation but at the cost of significantly increased training time and inference delay. Set-1 provides a middle ground, capturing sufficient local and global features while maintaining reasonable training and inference efficiency. These findings align with the quantitative results in Table 4, where Set-1 consistently achieves high validation accuracy with lower training loss compared to Set-0, without the excessive computational burden of Set-2.

The Tiny model with Set-1 hyper-parameters (512 sampled points, batch size of 16, 40 training epochs, and an initial learning rate of 1×10^{-3}) demonstrates a promising balance between computational efficiency and segmentation accuracy. The total training time for Set-1 is 1.54 minutes, which is only slightly higher than Set-0 (1.38 minutes) but significantly lower than Set-2 (2.44 minutes). This increase in training time is justified by the observed improvement in segmentation accuracy. The validation accuracy reaches 87.40%, surpassing both Set-0 (86.98%) and Set-2 (87.08%). The inference time per sample is 1.77 seconds, which is similar to Set-0 at 1.76s and faster than Set-2 at 1.86s. This suggests that Set-1 maintains efficient real-time performance while benefiting from improved model stability and accuracy. The choice of 512 points ensures that the model captures more geometric details than Set-0 with 256 points while avoiding the computational overhead of Set-2 with 1024 points.



(a) (b) (c)

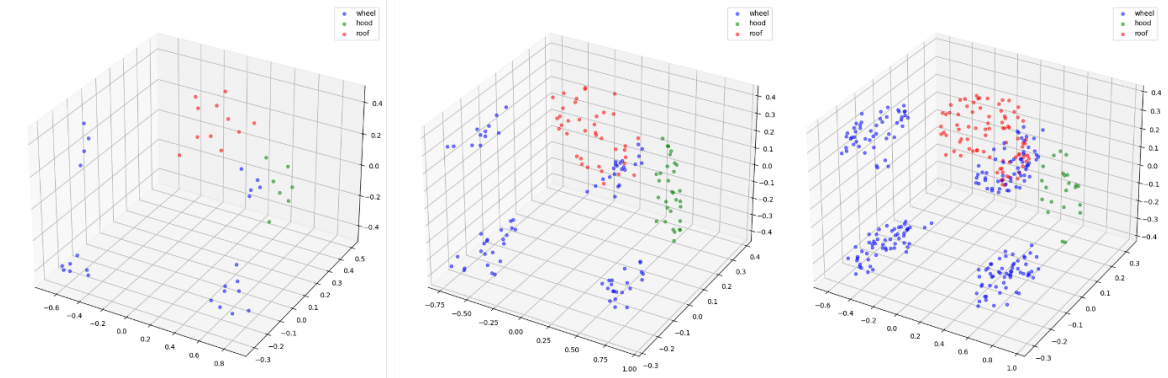
Figure 4. Medium model inference for airplane: a)Set-0 b)Set-1 c)Set-2.

4.2. Segmentation of Car

Training time per epoch is significantly higher with Set-2: 3.71s (Tiny) → 14.13s (Medium) → 33.22s (Large), making the Large model computationally expensive. Inference time per sample is stable in Set-1 and-2, however, it varies for Set-2, with the Large model performing inference the fastest (0.13s), possibly due to improved internal representations. The results for Set-1 portray better results in terms of validation accuracy. However, training and validation loss increase across models (8.56 – 8.56 for Tiny, 32.82 – 34.81 for Medium, 65.73 – 70.38 for Large), showing a higher complexity in optimization as model size increases. For Set-2 experiments, loss values skyrocket in the Medium and Large models (131.67 – 131.46 for Medium, 337.51 – 322.44 for Large), indicating overfitting and unstable training dynamics at this learning rate.

As seen in Table 5 and Figure 4, Set-1 achieves a well-balanced trade-off between segmentation accuracy and computational efficiency for the Car category. While Set-0 provides the shortest training time, it falls behind in accuracy. On the other hand, Set-2 produces detailed segmentation but comes with significantly increased training time and high training loss, especially in the Large model. Set-1 successfully captures both local and global features, making it the most practical choice.

Tiny model configurations struggle with overfitting and training stability, especially as batch size increases and the learning rate decreases in higher sets. Overall, the Tiny model's validation accuracy peaks in Set-1 (91.53%), indicating that a moderate batch size (16), increased training epochs (40), and lower learning rate (1×10^{-3}) provide the best generalization. Training and validation loss increase significantly from Set-0 (2.26 / 2.33) to Set-2 (33.96 / 33.77) on the Tiny model, suggesting that a higher learning rate (1×10^{-2}) in Set-0 helps maintain stable optimization. In terms of hyper-parameters sets, Set-1 strikes a balance, providing good validation accuracy with reasonable training efficiency. Set-0 offers fast performance but with limitations in terms of generalization for more complex models. Set-2 requires addressing the high training loss and the slow learning speed for the models.



(a) (b) (c)

Figure 5. Medium model inference for car: a)Set-0 b)Set-1 c) Set-2.

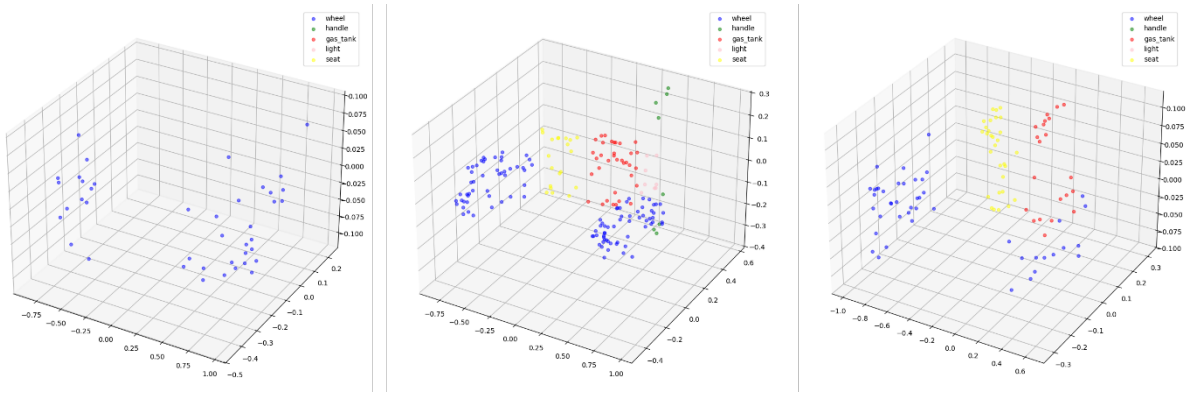
4.3. Segmentation of Motorbike

The Tiny model in Set-0 exhibits a relatively decent training accuracy (80.71%), but the validation accuracy (76.03%) is notably lower, suggesting some overfitting. This indicates that while the model performs well on the training data, it struggles to generalize effectively to unseen data. The training loss (2.4935) and validation loss (2.6193) are also on the lower side, indicating that the model is not overly complex. The Medium model in Set-0 demonstrates training accuracy (77.58%) and validation accuracy (67.99%), which is lower than expected (based on the results of the other classes). The training loss (8.7785) and validation loss (23.2788) show that the model could benefit from better hyperparameter tuning, as it might not be training efficiently with this configuration.

The Tiny model in Set-1 shows a small drop in both training accuracy (80.39%) and validation accuracy (75.19%), which may be a result of the higher training loss (8.5986) and validation loss (8.7146). The performance deterioration indicates a potential issue with the larger batch size and smaller learning rate, as it might have caused the model to struggle with the small number of samples. The Medium model in Set-1 performs slightly better with training accuracy (77.41%) and the best validation accuracy (71.38%), which indicates that the model is slightly more generalizable compared to its Set-0 counterpart. Validation loss (107.0709) is significantly higher than Set-0, but the improvement in validation accuracy suggests that the model has improved its performance with the new hyperparameters, particularly the batch size and learning rate.

The segmentation results obtained from the Medium dataset in Figure 6 show that Set-0 produces less refined segmentation boundaries, particularly in complex car models. In contrast, Set-2 generates more detailed segmentation but comes at the cost of significantly increased training time and inference delay. These findings align with the quantitative results in Table 5, where Set-1 consistently achieves high validation accuracy with lower training loss compared to Set-0 while avoiding the excessive computational burden of Set-2.

The Large model with Set-0 parameters shows train accuracy (73.64%) and validation accuracy (67.49%), with significantly higher train and validation loss compared to smaller models. With Set-1, the Large model shows a noticeable drop in validation accuracy (62.80%) compared to Set-0, with train accuracy (75.80%) still decent. The training loss (217.9921) and validation loss (675.8121) increase substantially, suggesting that the model's larger capacity is not necessarily translating to better generalization for the Motorbike class. This insight is confirmed in Set-1 experiments indicating that this model struggles with overfitting and does not generalize well on the Motorbike class data.



(a) (b) (c)

Figure 6. Medium model inference for motorbike: a)Set-0 b)Set-1 c)Set-2.

5. Discussion

Our findings demonstrated how architectural choices and hyper-parameter tuning impact segmentation accuracy, computational efficiency, and real-world usability. Choosing the appropriate model size and hyper-parameter configuration is crucial for practical applications. The Tiny Model with the Set-1 configuration, featuring a moderate batch size of 16, 512 sampled points, and an initial learning rate of 1×10^{-3}, provided the best balance between fast convergence, stable learning, and segmentation accuracy. Figure 7 summarizes the results in validation accuracy. The Tiny model across all the hyper-parameter sets achieves reasonably competitive performance, especially in the Car class where it attains the highest validation accuracy at Set-1 (91.53%).

However, the validation accuracy for Motorbike (around 75%) shows a significant drop when compared to the other classes. This may suggest that the model struggles more with the smaller dataset.

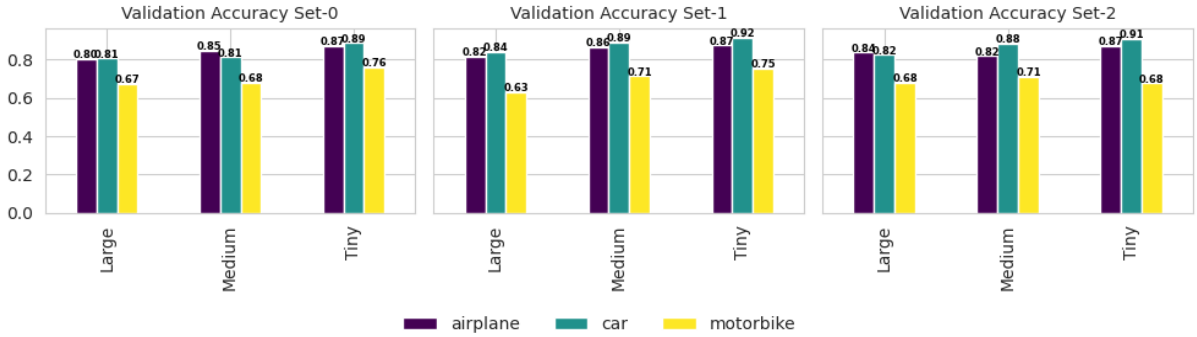


Figure 7. Validation accuracy across models and hyper-parameter sets.

The Airplane class and the Set-2 configuration of the Large Model, with larger sampled points (1024) and an extended training period (60 epochs), led to higher accuracy but at the expense of computational efficiency. However, it is computationally expensive, making it more practical for offline processing. For Car segmentation, the Tiny Model (Set-1) offers the best trade-off between accuracy and computational efficiency. Larger models introduce a higher computational burden without significant performance gains, making them less ideal for practical use. For the Motorbike class, with the smallest number of samples, Set-1 offers the best performance for both Medium and Large models, providing the most balanced trade-off between training time and validation accuracy. Set-2 needs careful adjustment to prevent overfitting and high loss, especially with Tiny and Medium models. Additionally, it is important to note that Tiny models may require further tuning, particularly in Set-2, to improve both training and validation accuracies. The observed trade-offs suggest that hyper-parameter selection should be tailored to the specific deployment constraints rather than simply optimizing for accuracy alone.

Our experimental results demonstrated that different model architectures exhibit distinct trade-offs between computational efficiency and segmentation accuracy. The Tiny model, with its shallower architecture, achieves competitive accuracy while maintaining low computational cost. This is primarily due to its ability to capture essential features efficiently without excessive parameter overhead. In our experiments, the Large model showed a higher risk of overfitting, particularly in the Motorbike class, where the dataset size was significantly smaller. This suggests that while deeper networks have greater capacity, they require careful regularization when handling limited data. Increasing the number of sampled points and batch size influences both convergence speed and generalization ability. The Set-1 configuration (512 sampled points, batch size of 16) provided the best balance, leading to stable learning dynamics and improved segmentation accuracy.

Future studies will explore the integration of semantic, instance, and panoptic segmentation into a unified model. This would enhance the adaptability of segmentation systems across diverse applications, particularly for autonomous perception tasks in autonomous vehicles, robotics, and geospatial analysis [7, 8]. Extending training datasets to include more diverse scenarios, such as adverse environmental conditions, varying object densities, and occlusions, will further improve model robustness [9, 19]. The ability to generalize across different real-world settings remains a key challenge in 3D vision research. Although this study briefly explored efficiency concerns, future research will focus on optimizing neural architectures to improve computational complexity without sacrificing segmentation accuracy. Approaches such as sparse convolutions, knowledge distillation, and quantization will be explored to enable faster and more efficient inference on edge devices [3, 7].

References

[1] Qi CR, Su H, Mo K, Guibas LJ. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. Proc IEEE Conf Comput Vis Pattern Recognit (CVPR), 2017; Honolulu, HI, USA. 652-660.

[2] Qi CR, Su H, Yi L, Guibas LJ. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. Adv Neural Inf Process Syst (NIPS), 2017; Long Beach, CA, USA. 30.

[3] Choy C, Gwak JY, Savarese S. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. Proc IEEE/CVF Conf Comput Vis Pattern Recognit (CVPR), 2019; Long Beach, CA, USA. 3075-3084.

[4] Behley J, Garbade M, Milioto A, Quenzel J, Behnke S, Stachniss C, Gall J. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. Proc IEEE/CVF Conf Comput Vis Pattern Recognit (CVPR), 2019; Long Beach, CA, USA. 9297-9307.
[5] Zhang Y, Zhou Z, David P, Yue X, Xi Z, Gong B, Foroosh H. PolarNet: An Improved Grid Representation for Online LiDAR Point Clouds Semantic Segmentation. Proc. IEEE/CVF Conf Comput Vis Pattern Recognit (CVPR), 2020; Virtual. 9601-9610.
[6] Cortinhal T, Tzelepis G, Aksoy EE. SalsaNext: Fast, Uncertainty-aware Semantic Segmentation of LiDAR Point Clouds for Autonomous Driving Int Symp Vis Comput., 2020; San Diego, CA, USA. 207-222.
[7] Tang H, Liu Z, Zhao S, Lin Y, Lin J, Wang H, Han S. Searching Efficient 3D Architectures with Sparse Point-Voxel Convolution Eur Conf Comput Vis (ECCV), 2020; Glasgow, UK. 685-702.
[8] Li S, Zhang C, He X. Shape-aware Semi-supervised 3D Semantic Segmentation for Medical Images. Med Image Comput Comput Assist Interv (MICCAI), 2020; Lima, Peru. 552–561.
[9] Wang Y, Sun Y, Liu Z, Sarma SE, Bronstein MM, Solomon JM. Dynamic Graph CNN for Learning on Point Clouds. ACM Trans Graph (TOG), 2019; 38 (5): 1-12.
[10] Graham B, Engelcke M, van der Maaten L. 3D Semantic Segmentation with Submanifold Sparse Convolutional Networks. Proc IEEE Conf Comput Vis Pattern Recognit (CVPR), 2018; Salt Lake City, UT, USA. 9224-9232.
[11] Thomas H, Qi CR, Deschaud JE, Marcotegui B, Goulette F, Guibas LJ. KPConv: Flexible and Deformable Convolution for Point Clouds. Proc. IEEE/CVF Int Conf Comput. Vis. (ICCV), 2019; Seoul, Korea (South). 6410-6419.
[12] Liu Z, Tang H, Lin Y, Han S. Point-Voxel CNN for Efficient 3D Deep Learning. Adv. Neural Inf. Process. Syst. (NeurIPS), 2019; Vancouver, Canada. 32.
[13] Heller N, Isensee F, Maier-Hein KH, Hou X, Xie C, Li F, Nan Y, Mu G, et al. The state of the art in kidney and kidney tumor segmentation in contrast-enhanced CT imaging: Results of the KiTS19 Challenge. Med Image Anal 2021; 67: 101821.
[14] Vinodkumar PK, Karabulut D, Avots E, Ozçınar C, Anbarjafari G. A Survey on Deep Learning Based Segmentation, Detection and Classification for 3D Point Clouds. Entropy, 2023; 25 (4): 635.
[15] He Y, Yu H, Liu X, Yang Z, Sun W, Anwar S, Mian A. Deep Learning Based 3D Segmentation: A Survey Inf Fusion 2025; 115: 102722.
[16] Chang AX, Funkhouser T, Guibas L, Hanrahan P, Huang Q, Li Z, Savarese S, Savva M, Song S, et al. ShapeNet: An Information-Rich 3D Model Repository. arXiv. 2015; 1512.03012.
[17] Xiang Y, Mottaghi R., Savarese S. Beyond PASCAL: A Benchmark for 3D Object Detection in the Wild. Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV), 2014; Steamboat Springs, CO, USA. 75-82.
[18] Jaderberg M, Simonyan K, Zisserman A, Kavukcuoglu K. Spatial Transformer Networks. Adv Neural Inf Process Syst (NIPS), 2015; Montreal, Canada. 29.
[19] Zhao H, Jiang L, Jia J, Torr P, Koltun V. Point Transformer. Proc. IEEE/CVF Int Conf Comput Vis (ICCV), 2021; Montreal, Canada. 16239-16248.