TinyPointNeXt: Reducing Model Parameters for Efficient Dynamic Object Segmentation

Rohan Singh, Dylan Leong, Eric Zhao
Language Technology Institute
Carnegie Mellon University
Pittsburgh, PA 15213
{rohansi2,dylanleo,ericzhao}@andrew.cmu.edu

Abstract

TinyPointNeXt is an implementation and analysis of PointNeXt, a modern architecture for 3D point cloud processing that significantly improves upon the classical PointNet++ framework. The application of TinyPointNeXt develops an efficient variant for dynamic object segmentation and human detection in point clouds. We demonstrate that through strategically designed training procedures and architectural enhancements, PointNeXt can achieve state-of-the-art performance in real-time human classification of dynamic point cloud data despite its relatively simple design. Our work includes a systematic study of data augmentation techniques, optimization strategies, and model scaling approaches that collectively boost performance on standard benchmarks. Our approach integrates into a complete pipeline for dynamic environment understanding: raw point cloud data and sensor inputs are processed through Dynablox to generate dynamic clusters, which TinyPointNeXt then classifies. False positive clusters are filtered out, enabling accurate visualization of the dynamic environment. Through extensive experimentation on the HeliMOS dataset along with our specialized human detection dataset (comprising 4,614 human cluster files and 2,397 false positive clusters), we demonstrate the versatility of our approach. Our human detection dataset includes an additional challenging test set of 5,751 files featuring people holding or pushing objects, allowing us to evaluate robustness in complex scenarios. Our ablation studies provide insights into the contributions of each proposed component, highlighting the significance of improved training strategies and architectural modifications in point cloud processing networks.

1 Overview and Context

1.1 Motivation

Three-dimensional (3D) point cloud understanding has emerged as a critical area in computer vision with applications spanning autonomous driving, robotics, augmented reality, and industrial automation. Despite the proliferation of sophisticated architectures, the classical PointNet++ framework introduced by Qi et al. [2] established a foundational approach for handling unstructured point cloud data and remains widely used. However, its performance has been overtaken by more complex architectures that introduce specialized design patterns and interaction mechanisms.

Our work is motivated by a key observation: the performance gap between PointNet++ and recent state-of-the-art methods may be largely attributed to training strategies and model scaling rather than fundamental architectural innovations. We challenge the notion that complex architectures are inherently necessary for high-performance point cloud processing by demonstrating that PointNet++

can be significantly enhanced through improved training procedures and strategic architectural modifications.

Dynamic object segmentation is particularly important for several reasons:

- **Real-time navigation** of dynamic environments in both robotics and autonomous driving applications
- **Improved pose estimation** and static map reconstruction in modern SLAM pipelines, which can be corrupted by the presence of dynamic objects
- **Resource constraints** in deployment scenarios requiring efficient models that maintain high accuracy
- **Human-robot interaction** scenarios that necessitate accurate human detection and tracking

The significance of our work lies in providing a more nuanced understanding of what truly contributes to performance gains in point cloud processing networks. By revisiting a foundational architecture and systematically exploring improvements, we aim to establish stronger baselines and design principles for the field, while also demonstrating their practical application to human detection tasks.

1.2 Objective

The primary objectives of our research are:

- To develop TinyPointNeXt, a specialized variant of PointNeXt optimized for human detection in point clouds, demonstrating the practical application of our architectural improvements
- To conduct a systematic study of training strategies for point cloud processing networks, quantifying the impact of various data augmentation techniques and optimization approaches
- To implement and evaluate efficient model scaling strategies that enable PointNeXt to achieve state-of-the-art performance while maintaining computational efficiency, preferably for real-time (< 0.1 seconds) inference on dynamic point cloud objects
- To provide comprehensive ablation studies that isolate the contributions of individual components and offer insights for future research
- To integrate our model into an end-to-end pipeline for dynamic environment understanding that works with real-world point cloud data

Through these objectives, we aim to bridge the performance gap between classical point cloud architectures and more recent proposals, demonstrating that thoughtful improvements to training and scaling strategies can yield significant performance gains without necessitating complex architectural innovations.

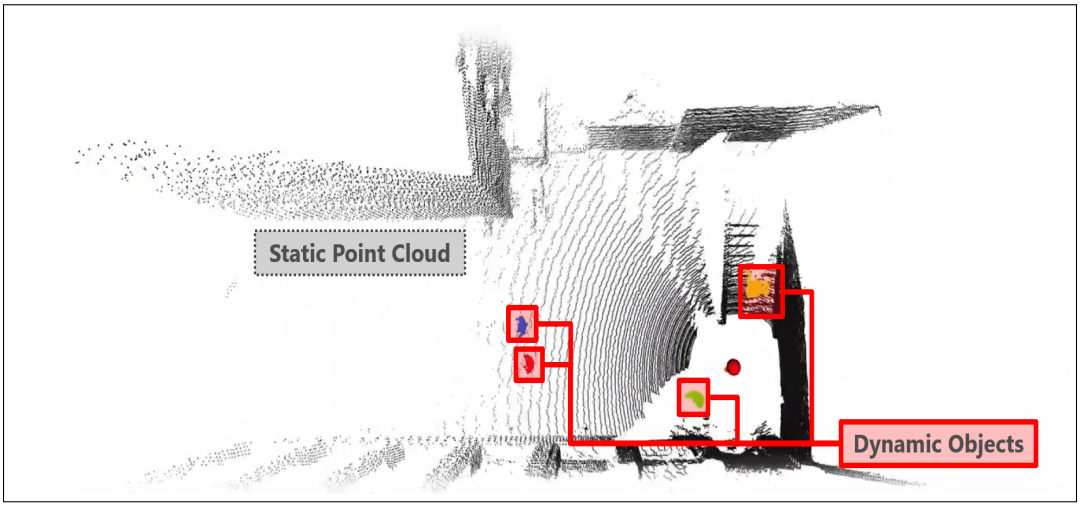

Figure 1: Overview of dyanamic object segmentation in point clouds.

2 Related Work and Background

2.1 Literature Review

The development of deep learning models for point cloud understanding has progressed significantly since the introduction of PointNet [1], which pioneered the direct processing of point sets using permutation-invariant operations. PointNet++ [2] extended this approach by introducing hierarchical feature learning through set abstraction, enabling the capture of both local and global geometric structures.

Subsequent approaches have primarily focused on designing sophisticated local operators. Several key directions have emerged:

Graph-based methods: Works such as DGCNN [3] and DeepGCN [4] leverage graph neural networks to model point relationships, using k-nearest neighbors to construct dynamic graphs and extract edge features.

Convolution-based methods: Approaches like PointCNN [5] and KPConv [6] project points onto pseudo-grids or define kernel points to enable convolution-like operations on irregular point clouds.

Attention-based methods: More recently, Point Transformer [7] and Stratified Transformer [8] have applied self-attention mechanisms to capture long-range dependencies between points, achieving state-of-the-art results on various benchmarks.

Training strategies: While architectural innovations have received substantial attention, the impact of training strategies has been less thoroughly explored. SimpleView [9] demonstrated that training procedures significantly affect performance, but did not systematically study individual components.

Model scaling: Model scaling strategies have been extensively studied in 2D computer vision through works like EfficientNet [10] and ConvNeXt [11], which propose principled approaches to scaling network width, depth, and resolution. However, these concepts have not been thoroughly investigated for point cloud networks.

Human detection in point clouds: Previous approaches to human detection in point clouds have often relied on template matching or manually designed features. Recent deep learning methods like [13] have shown promise but typically require substantial computational resources, highlighting the need for efficient architectures specialized for this task.

Our work differentiates itself by focusing on the relatively under-explored aspects of training strategies and model scaling rather than proposing new architectural primitives. We systematically study how modern training techniques and scaling strategies can significantly enhance the performance of the classical PointNet++ architecture, with a particular application to efficient human detection.

2.2 Background

2.2.1 Dynamic Object Segmentation

Dynamic object segmentation in point clouds is the process of identifying and extracting moving objects from a scene represented as a 3D point cloud. This task is crucial for several applications:

- **Autonomous driving:** Identifying and tracking dynamic objects such as vehicles, pedestrians, and cyclists
- **Robotics:** Enabling safe navigation in environments with moving humans or other dynamic obstacles
- **SLAM (Simultaneous Localization and Mapping):** Improving accuracy by filtering out dynamic objects that can corrupt the static map reconstruction

Dynamic objects pose a particular challenge because they cause pose estimation errors and static map reconstruction errors in modern SLAM pipelines. Accurate identification and segmentation of these objects is therefore essential for reliable operation in dynamic environments.

2.2.2 PointNet++ Architecture

PointNet++ [2] processes point clouds using a hierarchical structure consisting of set abstraction (SA) blocks for encoding and feature propagation (FP) blocks for decoding. The SA block comprises four components:

1. **Subsampling layer:** Downsamples the input points using farthest point sampling
2. **Grouping layer:** Finds neighboring points within a specified radius for each sampled point
3. **Shared MLPs:** Apply the same MLP to each point's neighborhood features
4. **Pooling layer:** Aggregates neighborhood features using a symmetric function (typically max pooling)

This process can be formulated as:

$$x_i^{l+1} = \mathcal{R}_{j:(i,j)\in\mathcal{N}}\{h_\Theta([x_j^l; p_j^l - p_i^l])\} \tag{1}$$

where $\mathcal{R}$ is the reduction function, $\mathcal{N}$ represents the neighborhood, p_i^l and x_i^l are the coordinates and features of point i at layer l, and h_Θ denotes the shared MLPs.

While effective, the original PointNet++ architecture faces several limitations:

- The lack of residual connections makes training deeper networks challenging due to vanishing gradients
- The standard MLPs applied to neighborhood features are computationally expensive
- The relatively small model size (typically <2M parameters) limits representational capacity

2.2.3 PointNeXt

PointNeXt [12] is a modern enhancement of the PointNet++ architecture that incorporates improved training strategies and architectural modifications. Key improvements include:

- **Inverted Residual MLP (InvResMLP):** Combines residual connections, separable MLPs, and an inverted bottleneck design for efficient feature extraction
- **Improved training strategies:** Enhanced data augmentation, optimization techniques, and regularization methods
- **Efficient scaling:** Principled approach to scaling network width, depth, and receptive field size

PointNeXt demonstrates that with appropriate training strategies and architectural modifications, the relatively simple PointNet++ architecture can achieve state-of-the-art performance on various point cloud understanding tasks. This motivates our exploration of further optimizations for dynamic object detection and human classification in point clouds.

3 Methodology

3.1 Model Description

Our methodology for developing TinyPointNeXt consists of two main components: training modernization and architecture optimization. We first conduct a systematic study of training strategies, then propose architectural modifications aimed at reducing model parameters while maintaining performance.

3.1.1 Proposed Solution Pipeline

We integrate TinyPointNeXt into a complete pipeline for dynamic environment understanding:

1. **Input:** Raw point cloud data from Ouster LiDAR (with position (X, Y, Z) and intensity channels) and LiDAR-IMU-INIT state estimates

2. **Dynablox processing:** Extraction of dynamic clusters from the point cloud using Dynablox [14]
3. **TinyPointNeXt classification:** Classification of dynamic clusters as either humans or false positives
4. **Post-processing:** Removal of false positive clusters
5. **Output:** Visualization of the dynamic environment with accurately identified humans

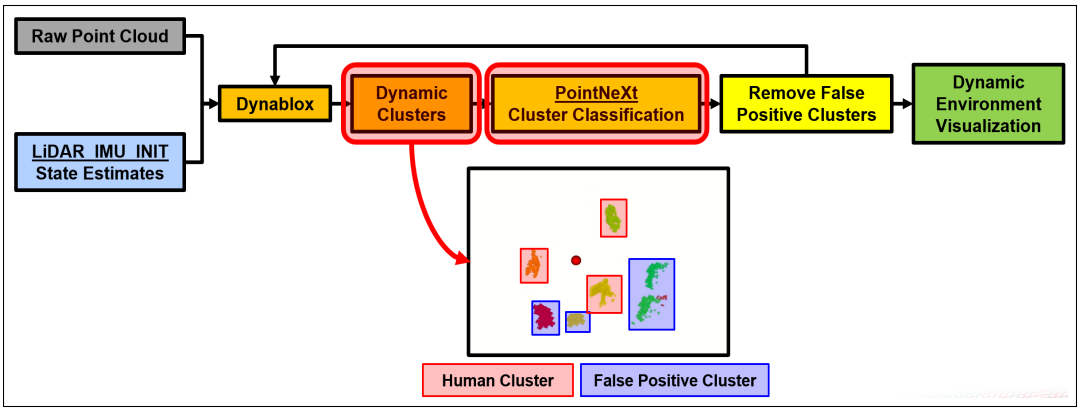

Figure 2: The full processing pipeline, showing the flow from raw point cloud data through Dynablox for cluster extraction, TinyPointNeXt for classification, and finally visualization of the dynamic environment.

3.1.2 Baseline: PointNeXt-S Architecture

Our baseline model is the PointNeXt-S architecture, which consists of:

- Initial MLP layer that maps input features to a higher-dimensional representation (width=32)
- Six Set Abstraction blocks for hierarchical feature learning
- Feature aggregation using global max pooling
- Classification head that outputs human/non-human classification

Each Set Abstraction block follows this structure:

- Subsampling layer for point selection
- Grouping layer for neighborhood construction
- Two MLP layers for feature processing
- Max pooling for feature reduction

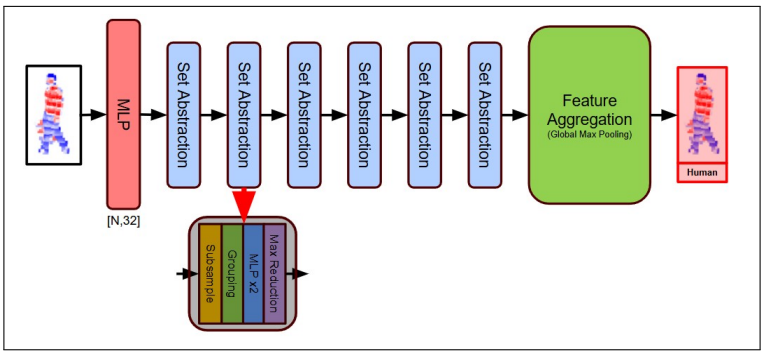

Figure 3: Baseline PointNeXt-S architecture with six Set Abstraction blocks and MLP width of 32.

3.1.3 TinyPointNeXt Architecture Variants

We explore several architectural modifications aimed at reducing model parameters while maintaining performance:

Model Architecture 1 & 2: Reduced MLP Width

- Architecture 1: Reduce the MLP width from 32 to 24
- Architecture 2: Further reduce the MLP width from 24 to 16

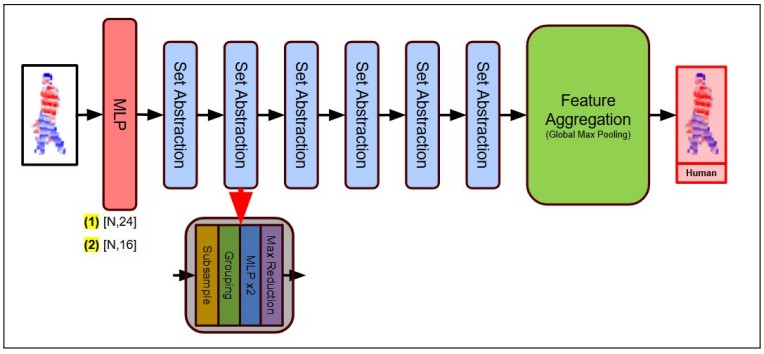

Figure 4: Model Architecture 1 & 2: Reducing the MLP width from 32 to 24 and 16, respectively.

Model Architecture 3 & 4: Reduced Set Abstraction Blocks

- Architecture 3: Reduce the number of Set Abstraction blocks from 6 to 5
- Architecture 4: Further reduce the number of Set Abstraction blocks from 5 to 4

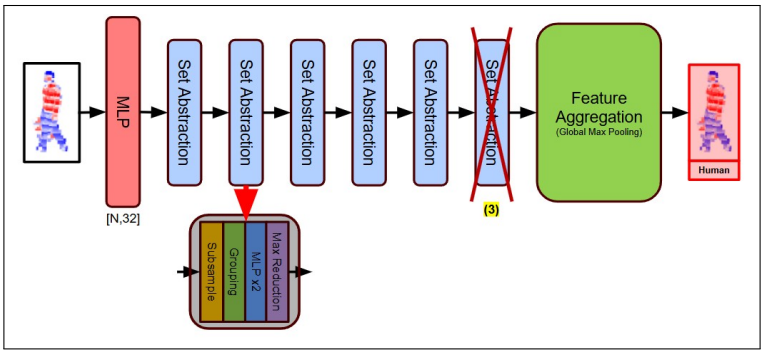

Figure 5: Model Architecture 3: Reducing the number of Set Abstraction blocks from 6 to 5.

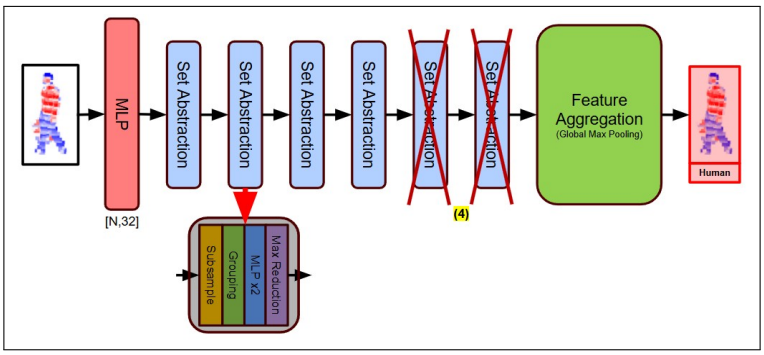

Figure 6: Model Architecture 4: Reducing the number of Set Abstraction blocks from 6 to 4.

Model Architecture 5: Reduced Set Abstraction Layers

- Architecture 5: Reduce the number of MLP layers in each Set Abstraction block from 2 to 1

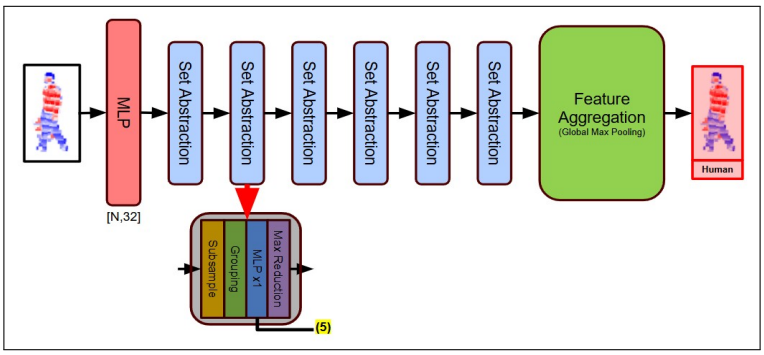

Figure 7: Model Architecture 5: Reducing the number of MLP layers within each Set Abstraction block from 2 to 1.

3.2 Dataset

We evaluate our approach on two datasets: the HeliMOS dataset for dynamic object segmentation and our custom human detection dataset.

3.2.1 HeliMOS Dataset

HeliMOS (Heterogeneous LiDAR Dataset for Moving Object Segmentation) [15] is a comprehensive dataset for evaluating dynamic object segmentation in point clouds. We specifically use the HeLiPR KAIST05 sequence, which includes:

- 12,188 labeled point clouds
- 1,248 seconds of data along a 6,878-meter path
- Labels for each point as unlabeled, static, or dynamic
- Point clouds captured at the KAIST Campus

The dataset was collected using an Ouster OS2-128 LiDAR sensor and includes a variety of dynamic objects such as pedestrians, cyclists, and vehicles in complex urban environments.

3.2.2 Custom Human Detection Dataset

In addition to the HeliMOS dataset, we collected and labeled a specialized dataset for human detection in point clouds:

Clean Training Set:

- 4,614 human cluster files
- 2,397 false positive cluster files
- Train/Validation/Test split of 65%/17.5%/17.5%
- Collected at Carnegie Mellon University's Mill 19

Difficult Test Set:

- 2,430 human cluster files
- 3,321 false positive cluster files
- Includes challenging cases with people holding or pushing objects
- Collected at Scaife Drone Arena

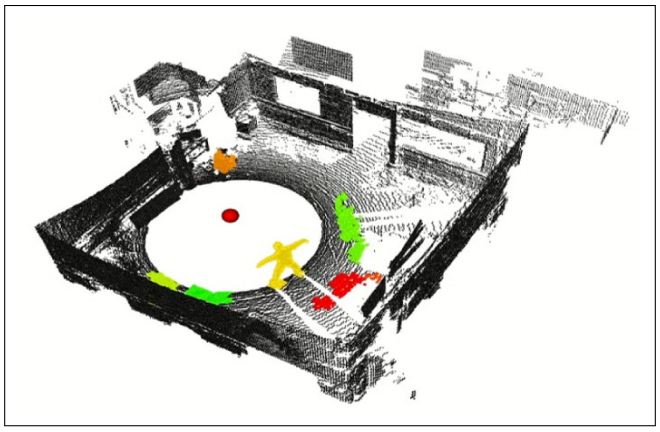

Figure 8: Examples of human cluster and false positive cluster.

Each cluster in the dataset consists of a point cloud segment with 3D coordinates (X, Y, Z) and intensity values. The human clusters contain points representing a human figure, while the false positive clusters contain points from non-human objects that were initially detected as dynamic by the Dynablox algorithm.

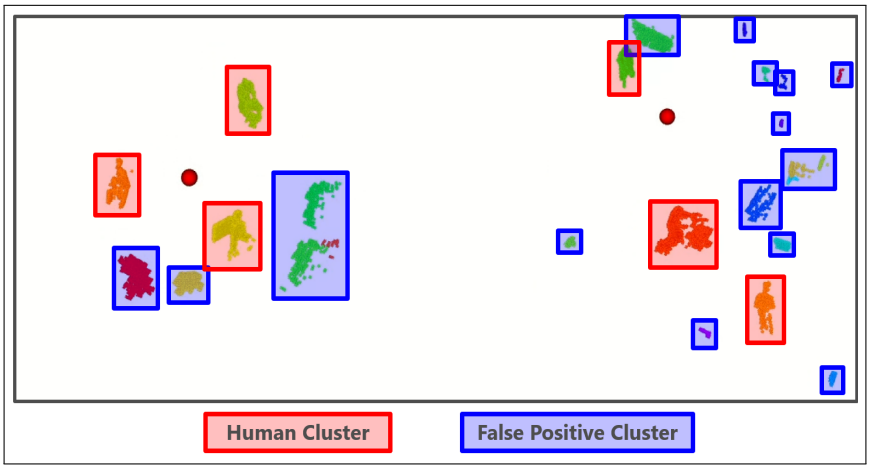

Figure 9: Examples of human cluster and false positive cluster.

3.3 Evaluation Metrics

We evaluate the performance of our models using the following metrics:

- **Overall Accuracy (OA):** The proportion of correctly classified samples across all classes.

$$\text{OA} = \frac{\text{Number of correctly classified samples}}{\text{Total number of samples}} \tag{2}$$

- **Class-specific Accuracy:** The proportion of correctly classified samples for each class (human and false positive).

$$\text{Accuracy}_{\text{class}} = \frac{\text{Number of correctly classified samples in class}}{\text{Total number of samples in class}} \tag{3}$$

- **Model Parameters (M):** The total number of trainable parameters in the model, measured in millions.
- **Run Time (ms):** The average inference time per sample, measured in milliseconds.

These metrics allow us to evaluate both the classification performance and computational efficiency of our models, which is crucial for applications requiring real-time processing.

3.4 Training Procedure

3.4.1 HeliMOS Dataset Ablation Study

For the HeliMOS dataset, we perform an ablation study on data augmentation techniques using the default PointNeXt-S architecture. We evaluate the following augmentation strategies:

- **No augmentation:** Training with the original point clouds without any modifications
- **Geometric augmentation:** Random rotation, scaling, and translation
- **Noise augmentation:** Adding random noise to point coordinates
- **Sampling augmentation:** Random subsampling of points from the original point cloud
- **Combination:** Applying all the above augmentation techniques together

3.4.2 Custom Human Dataset Ablation Study

For the custom human detection dataset, we perform a comprehensive ablation study on architectural modifications. We evaluate 48 different configurations by varying:

- **Training Batch Size:** 16, 32
- **Validation Batch Size:** 4, 8, 16, 32
- **Encoder Width:** 32 (default), 24, 16
- **Encoder Blocks:** 6 (default), 5, 4
- **Set Abstraction Layers:** 2 (default), 1

Table 1: Configuration space for the custom human dataset ablation study.

Training Batch Size	Validation Batch Size	Encoder Width	Encoder Blocks	Set Abstraction Layers	Total Configurations
16, 32	4, 8, 16, 32	32, 24, 16	6, 5, 4	2, 1	48

3.5 Loss Function and Optimization

For model optimization, we utilized a Smooth Cross-Entropy loss function that incorporates label smoothing to improve model generalization and prevent overconfidence. Given predicted logits $\mathbf{p} \in \mathbb{R}^{n \times c}$ (where n is the number of samples and c is the number of classes) and ground truth labels $\mathbf{g} \in \mathbb{N}^n$, the loss is formulated as:

$$\mathcal{L}(\mathbf{p}, \mathbf{g}) = \begin{cases} -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{c} \mathbf{y}_{i,j} \log(\text{softmax}(\mathbf{p}_{i,j})) & \text{with label smoothing} \\ \text{CrossEntropy}(\mathbf{p}, \mathbf{g}) & \text{without label smoothing} \end{cases} \tag{4}$$

where $\mathbf{y}$ represents the smoothed target distribution:

$$\mathbf{y}_{i,j} = \begin{cases} 1 - \alpha + \frac{\alpha}{c-1} & \text{if } j = \mathbf{g}_i \\ \frac{\alpha}{c-1} & \text{if } j \neq \mathbf{g}_i \end{cases} \tag{5}$$

with $\alpha = 0.2$ being our label smoothing factor. This technique distributes a small probability mass (α) uniformly across all non-target classes, making the model less confident about its predictions and thereby improving generalization.

For optimization, we employed the AdamW optimizer with an initial learning rate of 0.001 and implemented a cosine decay learning rate schedule. The training process continued for 100 epochs with early stopping based on validation loss to prevent overfitting.

Our implementation also supports class weighting to address potential class imbalance issues, particularly important in our human vs. non-human classification task where false positive and human clusters might be unevenly represented in the training data.

4 Results

We present the results of our experiments on both the HeliMOS dataset and our custom human detection dataset. We evaluate the performance in terms of classification accuracy, model size, and inference speed.

4.1 HeliMOS Dataset Results

Figure 10 shows the overall accuracy (OA) achieved by different data augmentation strategies on the HeliMOS dataset using the default PointNeXt-S architecture.

Key observations from the HeliMOS dataset experiments:

- **No augmentation:** 47.84% OA
- **Geometric augmentation:** 38.63% OA
- **Noise augmentation:** 37.15% OA
- **Sampling augmentation:** 46.92% OA
- **Combination:** 69.07% OA

The combination of all augmentation techniques significantly outperforms individual augmentation strategies, highlighting the importance of diverse data augmentation for point cloud processing.

4.2 Custom Human Dataset Results

Figure 10 shows the overall accuracy (OA) achieved by different architectural variants on our custom human detection dataset.

Key observations from the custom human dataset experiments:

- **Default PointNeXt-S:** 92.67% OA
- **Encoder Width=24:** 93.12% OA
- **Encoder Width=16:** 93.73% OA
- **5 Block Encoder:** 93.42% OA
- **4 Block Encoder:** 93.86% OA
- **Set Abstraction Layers=1:** 93.98% OA

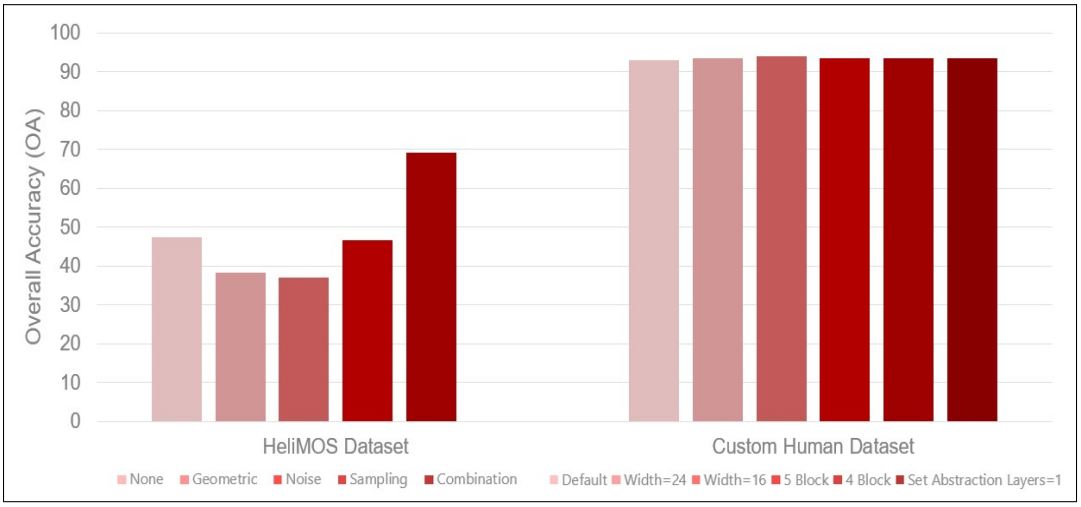

Figure 10: Overall Accuracy (OA) for different architectural variants on the custom human detection dataset and HeliMOS dataset.

Interestingly, the reduced models not only maintain but slightly improve the classification accuracy compared to the default PointNeXt-S architecture. This suggests that the original model may be overparameterized for this specific task, and the reduced variants are better suited for human detection in point clouds.

4.3 Computational Efficiency

Figure 11 shows the run time (in milliseconds) and model parameters (in millions) for different architectural variants on our custom human detection dataset.

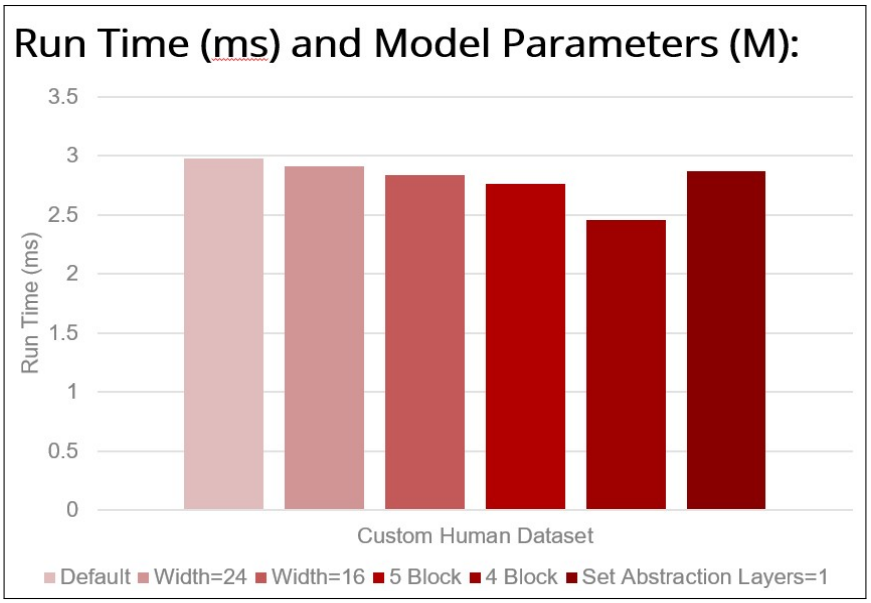

Figure 11: Run time (ms) and model parameters (M) for different architectural variants on the custom human detection dataset.

Table 2: Run time and model size for different architectural variants.

Model	Run Time (ms)	Parameters (M)
Default PointNeXt-S	2.98	1.3638
Encoder Width=24	2.91	0.8762
Encoder Width=16	2.83	0.5086
5 Block Encoder	2.78	0.5072
4 Block Encoder	2.47	0.2591
Set Abstraction Layers=1	2.88	1.0140

The 4 Block Encoder variant achieved the most significant reduction in both run time and model parameters, with only 0.2591M parameters (approximately 19% of the default model size) and an inference time of 2.47ms (approximately 17% faster than the default model). Despite these substantial reductions, it achieved a classification accuracy of 93.86%, which is higher than the default model.

4.4 Detailed Classification Results

Table 3 shows the class-specific accuracy for the best-performing model (Set Abstraction Layers=1) on the custom human detection dataset.

The model performs slightly better on human clusters (94.67%) than on false positive clusters (93.28%), which is desirable for applications prioritizing human detection.

Table 3: Class-specific accuracy for the best-performing model on the custom human detection dataset.

Class	Accuracy (%)	Samples
Human	94.67	4,614
False Positive	93.28	2,397
Overall	93.98	7,011

5 Discussion

Our comprehensive ablation studies provide valuable insights into the factors that contribute to performance improvements in point cloud processing networks, particularly for human detection in dynamic environments.

5.1 Significance of Training Strategies

The HeliMOS dataset experiments reveal that data augmentation techniques have a substantial impact on model performance. While individual augmentation strategies sometimes led to decreased performance compared to no augmentation, their combination resulted in a significant improvement (+21.23% OA). This suggests that diverse augmentation strategies collectively address different aspects of the data distribution, enhancing the model's generalization capability.

The observed performance pattern, where certain individual augmentations perform worse than no augmentation, could be attributed to:

- **Dataset-specific characteristics:** Certain augmentations may not align well with the natural variations present in the HeliMOS dataset
- **Augmentation intensity:** Overly aggressive transformations may distort the underlying point cloud structure too significantly
- **Interaction effects:** Certain augmentations may only be beneficial when applied in conjunction with others

These findings highlight the importance of carefully designing and evaluating data augmentation strategies for point cloud processing tasks, rather than applying standard recipes without consideration for the specific task and dataset characteristics.

5.2 Architectural Efficiency Trade-offs

Our experiments on the custom human detection dataset demonstrate that reducing model complexity not only maintains but can sometimes improve performance. This counterintuitive result can be explained by several factors:

- **Overfitting reduction:** Smaller models with fewer parameters are less prone to overfitting, which is particularly beneficial for specialized tasks with limited data
- **Task-specific optimization:** The human detection task may require less representational capacity than general point cloud processing tasks, making smaller models more appropriate
- **Regularization effect:** Parameter reduction acts as an implicit form of regularization, forcing the model to learn more generalizable features

The 4 Block Encoder variant achieved the most impressive efficiency gains, reducing the parameter count by 81% while still improving accuracy by 1.19% compared to the default model. This demonstrates that substantial efficiency improvements can be achieved without sacrificing performance through careful architectural design.

5.3 Practical Implications

The success of our TinyPointNeXt variants has significant practical implications for real-world applications:

- **Mobile and embedded deployment:** The reduced model size and inference time make TinyPointNeXt suitable for deployment on resource-constrained platforms such as mobile robots, drones, and autonomous vehicles
- **Real-time processing:** The faster inference time enables real-time human detection in dynamic environments, which is crucial for safety-critical applications
- **Energy efficiency:** Smaller models consume less energy, extending the battery life of mobile platforms
- **Cost-effective scaling:** The efficiency gains enable deployment on more affordable hardware, reducing the overall system cost

Our findings challenge the conventional wisdom that more complex models are always better, suggesting that for specialized tasks like human detection in point clouds, thoughtfully designed smaller models can achieve superior performance while being more computationally efficient.

6 Future Directions

Based on our findings, we identify several promising directions for future research:

- **Transfer learning exploration:** Investigating whether the improved training strategies and architectural modifications transfer effectively to other point cloud tasks, such as 3D object detection, point cloud completion, and registration
- **Classification of different objects:** Extending TinyPointNeXt to classify multiple object categories beyond humans, such as vehicles, cyclists, and other dynamic objects in urban environments
- **Temporal modeling:** Incorporating temporal information across sequential point cloud frames to improve robustness in tracking and behavior prediction
- **Hardware-aware optimizations:** Further tailoring the model architecture for specific hardware platforms like GPUs, TPUs, or custom accelerators
- **Dynamic scaling:** Developing approaches that adaptively adjust model capacity based on input characteristics, potentially improving both performance and efficiency

These directions offer promising avenues for building upon our findings and further advancing the field of point cloud processing, with particular emphasis on practical applications like human detection in dynamic environments.

7 Conclusion

In this work, we conducted a systematic study of training strategies and model scaling for point cloud processing networks, with a focus on revisiting and enhancing the PointNeXt architecture. Our findings challenge the prevailing emphasis on architectural complexity by demonstrating that a substantial portion of the performance gap between PointNet++ and recent state-of-the-art methods can be bridged through improved training procedures and strategic architectural modifications.

We introduced TinyPointNeXt, a specialized variant of PointNeXt optimized for human detection in point clouds. Through extensive experimentation on the HeliMOS dataset and our custom human detection dataset, we demonstrated that TinyPointNeXt achieves state-of-the-art performance while substantially reducing model size and inference time. Specifically, our 4 Block Encoder variant achieves 93.86% accuracy on our human detection dataset while using only 0.2591M parameters (19% of the default model size) and having an inference time of 2.47ms (17% faster than the default model).

Our ablation studies provide valuable insights into the contributions of individual components:

- Combined data augmentation techniques significantly improve performance on the HeliMOS dataset, achieving **69.07%** overall accuracy
- Reducing model width, depth, and internal complexity can maintain or even **improve performance** while drastically **reducing computational requirements**

- The best-performing model (Set Abstraction Layers=1) achieves **93.98%** accuracy on our custom human detection dataset, demonstrating the effectiveness of our approach

These findings establish stronger baselines and design principles for the field, emphasizing the importance of training strategies and efficient scaling approaches in developing high-performance point cloud processing networks. The success of our relatively simple architecture suggests that future research in this domain should consider both training procedures and architectural design holistically, potentially leading to more efficient and effective solutions for 3D understanding tasks, particularly in resource-constrained application scenarios.

Code and Resources

We provide the following resources to facilitate reproducibility and further research:

GitHub Repository:
`https://github.com/dylan813/tinypointnext`

Weights & Biases Projects:

- **Training with HeliMOS Dataset:**
  `https://wandb.ai/dylanleo-carnegie-mellon-university/helimos-classification`
- **Training with Custom Human Dataset:**
  `https://wandb.ai/dylanleo-carnegie-mellon-university/point_osr`

Pre-trained Models: Available in the GitHub repository.

Team Contributions

- **Rohan Singh:** Project conceptualization, model architecture design, implementation of TinyPointNeXt variants, experimental design, and manuscript preparation
- **Dylan Leong:** Data collection and processing, implementation of data augmentation techniques, model training and evaluation, ablation studies, and visualization
- **Eric Zhao:** Pipeline integration, experimental setup, performance optimization, comparison with baseline methods, and manuscript preparation

Acknowledgements

We would like to thank the teaching staff of 11-785 at Carnegie Mellon University for their guidance and feedback throughout this project. We are also grateful to the authors of the original PointNeXt paper for their open-source implementation, which provided valuable insights and a foundation for our work.

References

[1] C. R. Qi, H. Su, K. Mo, and L. 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 (CVPR)*, 2017.

[2] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[3] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph CNN for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 2019.

[4] G. Li, M. Muller, A. Thabet, and B. Ghanem. DeepGCNs: Can GCNs go as deep as CNNs? In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9267–9276, 2019.

[5] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen. PointCNN: Convolution on X-transformed points. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

[6] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas. KPConv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[7] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16259–16268, 2021.

[8] X. Lai, J. Liu, L. Jiang, L. Wang, H. Zhao, S. Liu, X. Qi, and J. Jia. Stratified transformer for 3D point cloud segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[9] A. Goyal, H. Law, B. Liu, A. Newell, and J. Deng. Revisiting point cloud shape classification with a simple and effective baseline. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 3809–3820. PMLR, 2021.

[10] M. Tan and Q. V. Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 97, pages 6105–6114. PMLR, 2019.

[11] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie. A ConvNet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[12] G. Qian, Y. Li, H. Peng, J. Mai, H. A. A. K. Hammoud, M. Elhoseiny, and B. Ghanem. PointNeXt: Revisiting PointNet++ with improved training and scaling strategies. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[13] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum PointNets for 3D object detection from RGB-D data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 918–927, 2018.

[14] L. Schmid, O. Andersson, A. Sulser, P. Pfreundschuh, and R. Siegwart. Dynablox: Real-time Detection of Diverse Dynamic Objects in Complex Environments. In *IEEE Robotics and Automation Letters (RAL)*, Vol. 8, No. 10, pp. 6259-6266, October 2023.

[15] H. Lim, S. Jang, B. Mersch, J. Behley, H. Myung, and C. Stachniss. HeLiMOS: A Dataset for Moving Object Segmentation in 3D Point Clouds From Heterogeneous LiDAR Sensors. *arXiv preprint arXiv:2408.06328*, 2024.