Electronics and Information Technologies, 2026, 35, 45–56
http://publications.lnu.edu.ua/collections/index.php/electronics/index

UDC: 004.93

SEGMENTATION OF COMPLEX STRUCTURES IN POINT CLOUDS USING DEEP NEURAL NETWORKS

Viktor Vdovychenko* & ***Yuriy Furgala***
Faculty of Electronics and Computer Technologies
Ivan Franko National University of Lviv,
50 Drahomanova Street, Lviv, 79005, UKRAINE

Vdovychenko V. M., & Furgala Y. M. (2026). Segmentation of Complex Structures in Point Clouds Using Deep Neural Networks. *Electronics and Information Technologies*, 35, 45–56. https://doi.org/10.30970/eli.35.4

ABSTRACT

Background. Point cloud part segmentation assigns a semantic part label to every point and provides a more detailed description of object geometry than object-level classification. Architecture selection remains difficult because models with similar aggregate benchmark scores may differ substantially in category-level reliability and computational cost.

Materials and Methods. This study compares PointNet++, Dynamic Graph Convolutional Neural Network (DGCNN), and Point Transformer on the ShapeNet Part benchmark, which represents sixteen object categories with fifty part labels. Final evaluation was performed on 2,874 test shapes represented by 2,048 points. Segmentation quality was assessed using point accuracy, class-average accuracy, class mean intersection over union (mIoU), and instance mean intersection over union (mIoU). A separate batch-size-one benchmark measured inference latency, throughput, and graphics memory use.

Results and Discussion. Point Transformer achieved the highest overall quality, with 94.4% point accuracy, 87.7% class-average accuracy, 82.9% class mIoU, and 85.7% instance mIoU. DGCNN reached 85.3% instance mIoU while requiring only 15.87 ms per shape, compared with 53.39 ms for Point Transformer and 344.62 ms for PointNet++. Aggregate differences were small, but category-level results were more pronounced. Point Transformer produced the highest mIoU in eight of sixteen categories, while DGCNN and PointNet++ led in five and three categories, respectively. The largest architecture-dependent differences were observed for Motorbike, Earphone, and Rocket, demonstrating that similar overall scores can conceal distinct failure patterns.

Conclusion. No architecture dominated all practical criteria. Point Transformer provided the strongest segmentation quality, whereas DGCNN offered the most favorable measured speed-quality trade-off. PointNet++ remained competitive in quality but showed substantially higher latency in the evaluated implementation. Model selection should therefore consider category-level behavior, input configuration, and computational constraints together with aggregate mIoU.

Keywords: point cloud segmentation, part segmentation, ShapeNet Part, PointNet++, DGCNN, Point Transformer

INTRODUCTION

Point clouds are a fundamental representation of three-dimensional geometry and are widely used in robotics, autonomous systems, industrial inspection, digital twins, reconstruction, and geometric analysis [1]. Unlike raster images, point clouds are unordered and irregularly sampled, and they do not provide a fixed neighborhood structure.

© 2026 Viktor Vdovychenko & Yuriy Furgala. Published by the Ivan Franko National University of Lviv on behalf of Електроніка та інформаційні технології / Electronics and Information Technologies. This is an Open Access article distributed under the terms of the Creative Commons Attribution 4.0 License which permits unrestricted reuse, distribution, and reproduction in any medium, provided the original work is properly cited.

These properties make direct processing by conventional grid-based convolution difficult and motivate neural architectures designed specifically for point sets.

Object-level classification assigns a single semantic label to an entire point cloud, whereas part segmentation assigns a label to every point. The latter therefore provides a substantially more detailed representation of an object's internal structure and is directly relevant to applications that require localization of functional or geometric parts, extraction of dimensions, or subsequent estimation of physical characteristics. PointNet demonstrated that point sets can be processed directly with permutation-invariant operations and established a common foundation for both classification and segmentation [2].

Modern point-cloud networks use different mechanisms to capture local and global geometric relationships. PointNet++ introduces hierarchical sampling, neighborhood grouping, and local PointNet operations to learn features at multiple spatial scales [3]. DGCNN represents points as a dynamically reconstructed graph and applies EdgeConv to learn relationships between each point and its local neighbors in feature space [4]. Point Transformer adapts self-attention to point sets and incorporates relative position information into feature aggregation [5]. These architectural differences can influence not only segmentation quality but also computational cost and sensitivity to individual object categories.

Recent research continues to expand transformer-based and segmentation-oriented point-cloud processing. Point Transformer V3 emphasizes scalable and efficient serialized processing [8], RangeViT applies transformer representations to large-scale LiDAR semantic segmentation [9], and CamPoint explores visibility-based local and global context modeling for point cloud segmentation [10]. Despite this progress, classic architectures such as PointNet++, DGCNN, and the original Point Transformer remain useful reference points because they represent three distinct and widely used design principles: hierarchical point aggregation, dynamic graph learning, and position-aware attention.

Direct comparison of results reported in independent studies is difficult because implementations commonly differ in the number of sampled points, input features, data augmentation, optimizer settings, voting strategies, checkpoint selection, and hardware platforms. Small differences in aggregate mean Intersection over Union (mIoU) may therefore be difficult to interpret without a controlled benchmark. Moreover, an architecture that has the best average score may not be preferable when latency, model size, or category-level reliability are important deployment constraints.

The aim of this study is to compare PointNet++, DGCNN, and Point Transformer for object part segmentation on the ShapeNet Part benchmark and to analyze their segmentation quality, training dynamics, category-level behavior, and computational efficiency. The comparison uses a common final test protocol with 2,048 points per shape, 2,874 test objects, batch size one, and the same graphics processing unit. Because the reference implementations do not expose identical input pipelines, the comparison also explicitly reports implementation-specific differences. The study is intended to identify practical trade-offs that are not visible from a single aggregate mIoU value.

MATERIALS AND METHODS

Dataset

Experiments were conducted on ShapeNet Part, a benchmark derived from the ShapeNet collection and designed for fine-grained part-level annotation of three-dimensional objects [6, 7]. The complete benchmark contains 16,881 shapes from 16 object categories with 50 semantic part labels in total. Each object category contains its own subset of valid part labels. The categories evaluated in this study were Airplane, Bag, Cap, Car, Chair, Earphone, Guitar, Knife, Lamp, Laptop, Motorbike, Mug, Pistol, Rocket, Skateboard, and Table. Representative examples of ShapeNet Part objects with ground-truth part annotations are shown in **Fig. 1**.

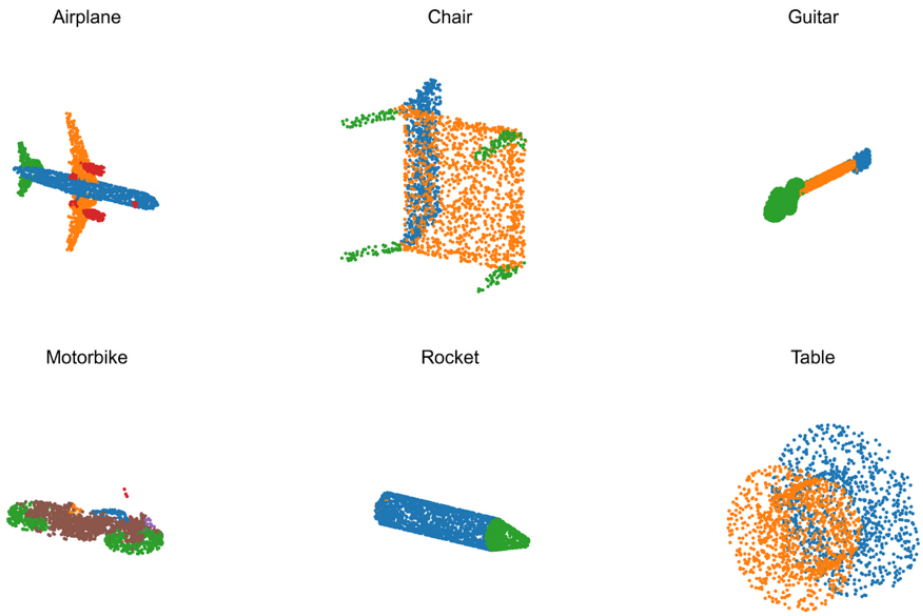


Fig. 1. Representative ShapeNet Part objects with ground-truth part segmentation

The final benchmark used 2,874 test shapes, and every shape was represented by 2,048 points. The evaluated implementations used the standard ShapeNet Part training/test organization provided by their respective repositories, while the final quantitative comparison reported in this paper was recomputed on the common 2,874-object test set. This distinction is important because repository-specific preprocessing and training pipelines may not be identical even when the same benchmark is used.

Evaluated Architectures

PointNet++. PointNet++ extends PointNet by constructing a hierarchical representation of a point cloud. Farthest point sampling selects representative centroids, local neighborhoods are grouped around these points, and PointNet-style functions encode increasingly abstract features [3].

DGCNN. Dynamic Graph Convolutional Neural Network constructs a k-nearest-neighbor graph and applies EdgeConv to learn local relationships between a point and neighboring features [4]. The graph is recomputed in feature space across layers, allowing local connectivity to adapt as representations evolve. The evaluated part-segmentation configuration used k = 40, embedding dimension 1,024, 2,048 points.

Point Transformer. Point Transformer applies position-aware self-attention to point clouds. Relative spatial information is incorporated into attention-based feature aggregation so that the network can model both geometric and feature relationships [5].

Training and Evaluation Protocol

All three models were trained from scratch using their reference part-segmentation implementations. PointNet++ was trained for 250 epochs with batch size 16 and the Adam optimizer. DGCNN was trained for 250 epochs with batch size 8, stochastic gradient descent, momentum 0.9, a cosine learning-rate schedule, and dropout 0.5. Point Transformer was trained for 250 epochs using its SGD configuration. The best checkpoints were selected during training according to the validation criteria implemented in the corresponding repositories.

The experiments were performed using publicly available PyTorch implementations of the three architectures. For reproducibility, the exact source repositories used in this study were: PointNet++ [11]; DGCNN [12]; and Point Transformer (unofficial implementation) [13]. The reported inference latency and throughput therefore characterize these specific implementations and software revisions rather than architecture-independent theoretical performance.

For the final benchmark, every model processed 2,048 points per object with batch size one. Twenty warm-up batches were excluded before latency measurements. The benchmark evaluated all 2,874 test shapes and reported point accuracy, class-average accuracy, class-average mIoU, instance-average mIoU, latency, throughput, memory consumption, number of parameters, and checkpoint size. PointNet++ was evaluated with one vote to avoid introducing test-time ensembling into the comparison.

The final inference protocol was standardized where possible. Therefore, the results should be interpreted as a comparison of the evaluated implementations rather than as an ablation study that isolates architectural differences while keeping all input features and preprocessing steps identical. The main training and evaluation settings for the three models are summarized in **Table 1**.

Table 1. **Experimental configuration of the evaluated segmentation models**

Parameter	PointNet++	DGCNN	Point Transformer
Input points	2048	2048	2048
Training epochs	250	250	250
Training batch size	16	8	16
Optimizer	Adam	SGD	SGD
Final test batch size	1	1	1
Test shapes	2874	2874	2874

Evaluation Metrics

Point accuracy was computed as the proportion of point labels predicted correctly over all points in the test set. Class-average accuracy gives equal importance to the semantic part classes and is therefore more sensitive to less frequent labels than overall point accuracy.

Intersection over Union (IoU) was used as the principal segmentation measure. For each part, IoU quantifies the overlap between the predicted and ground-truth point sets relative to their union. Class mIoU averages segmentation quality across the 16 object categories, whereas instance mIoU averages the part IoU over individual test shapes. Reporting both metrics is important because category frequencies are not uniform and strong performance on common categories can otherwise dominate an instance-weighted score.

Category-level mIoU was additionally analyzed for all 16 ShapeNet Part categories. This analysis was used to identify cases in which similar aggregate scores conceal architecture-specific strengths or weaknesses. For selected categories discussed in detail, variability across individual test shapes was additionally characterized by the standard deviation of per-shape IoU and is reported as ± SD.

Computational Efficiency Evaluation

Computational efficiency was measured using a separate inference benchmark with batch size one. Mean latency was expressed in milliseconds per shape, and throughput

was calculated as processed shapes per second. Graphics processing unit memory was reported as maximum allocated and reserved memory during the benchmark.

Latency measurements excluded data loading and result saving and were collected after 20 warm-up batches. The use of batch size one reflects a low-latency inference scenario and makes the measurements directly comparable across the three benchmark scripts.

Experimental Environment

The experiments were conducted on a workstation equipped with an NVIDIA GeForce GTX 1080 Ti graphics processing unit with 11 GB of video memory, an Intel Core i5-10600K processor, and 32 GB of random-access memory. The segmentation experiments were executed under Ubuntu 24.04 using Python-based PyTorch implementations [14]. Separate software environments were used where required by repository-specific dependencies, while the final inference benchmark used the same graphics processing unit for all models.

RESULTS AND DISCUSSION

Overall Segmentation Performance

Table 2 summarizes the final segmentation quality obtained on the common ShapeNet Part test set. All three architectures achieved point accuracy above 94% and instance mIoU above 85%, indicating that the overall performance differences are relatively small. Point Transformer achieved the strongest aggregate result, with 94.4% point accuracy, 87.7% class-average accuracy, 82.9% class mIoU, and 85.7% instance mIoU. DGCNN obtained 94.3% point accuracy and 85.3% instance mIoU, while PointNet++ achieved 94.2% and 85.1%, respectively.

Table 2. **Overall segmentation performance on the ShapeNet Part test set**

Model	Point accuracy, %	Class avg. accuracy, %	Class mIoU, %	Instance mIoU, %
PointNet++	94.2	87.2	81.8	85.1
DGCNN	94.3	84.2	81.4	85.3
Point Transformer	94.4	87.7	82.9	85.7

The ranking differs slightly depending on the selected metric. PointNet++ produced higher class-average accuracy than DGCNN, whereas DGCNN produced higher instance mIoU. The lower class-average accuracy of DGCNN compared with PointNet++ and Point Transformer suggests that its errors are distributed less uniformly across semantic part classes. Because class-average accuracy gives equal weight to each part class, weaker performance on less frequent or more difficult part labels can reduce this metric even when instance-level mIoU remains competitive. This indicates that a single aggregate measure is not sufficient to characterize segmentation behavior. The maximum difference in instance mIoU between the three models was only 0.6 percentage points, but the class mIoU difference between DGCNN and Point Transformer reached approximately 1.5 percentage points. The category-level analysis below shows that these apparently small aggregate differences are composed of much larger gains and losses on individual object types.

Training Dynamics

Fig. 2 compares test accuracy and instance mIoU over the training trajectories. All three networks improved rapidly during the initial epochs and then entered a slower

refinement stage. PointNet++ reached strong segmentation quality comparatively early, and its best checkpoint used in the benchmark was obtained at epoch 136. Point Transformer continued improving over a longer interval, with the benchmark checkpoint selected at epoch 170. DGCNN showed stronger epoch-to-epoch fluctuations in the test curves, including several temporary drops, but recovered and converged to an instance mIoU comparable with the other architectures.

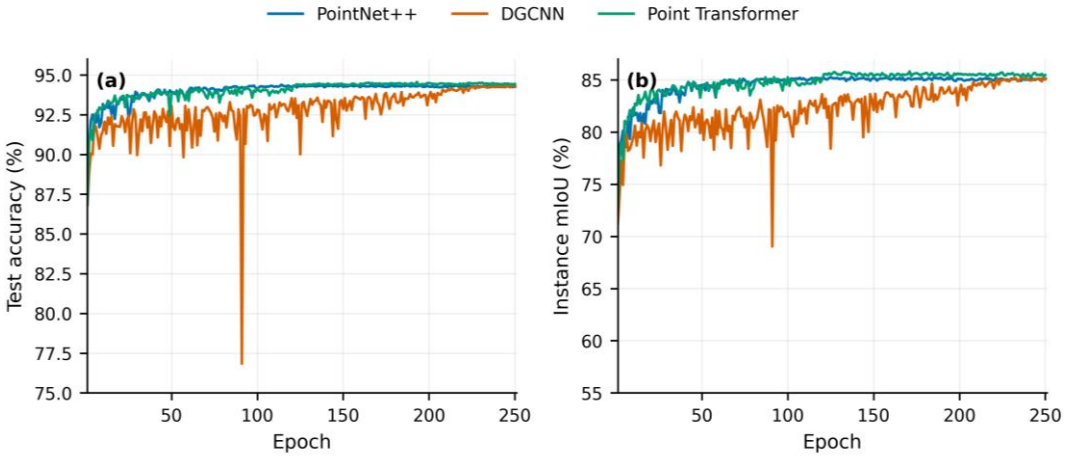


Fig. 2. Training dynamics of the evaluated segmentation models: (a) test accuracy and (b) instance mIoU

The training curves should be interpreted as optimization behavior rather than as a direct ranking of generalization quality. The final benchmark was recomputed from saved checkpoints under a separate batch-size-one protocol; therefore, the values in **Table 2** are the primary basis for the final comparison. Nevertheless, the curves demonstrate that architectures with similar final mIoU can reach that level through different convergence trajectories.

Per-Category Segmentation Performance

Fig. 3 presents mIoU for each of the 16 ShapeNet Part categories. Point Transformer achieved the highest category mIoU in eight categories, DGCNN in five, and PointNet++ in three. The results therefore support the aggregate ranking of Point Transformer while also showing that no model dominates every object type.

The largest architecture-dependent difference was observed for Motorbike. DGCNN reached 61.3 ± 10.1% mIoU, PointNet++ 71.2 ± 9.8%, and Point Transformer 73.5 ± 10.0%, giving a spread of more than 12 percentage points. Earphone also revealed a clear weakness of PointNet++, which achieved 69.9 ± 17.3% compared with 77.6 ± 18.8% for DGCNN and 78.6 ± 17.2% for Point Transformer. The relatively large standard deviations indicate substantial variation in segmentation quality across individual objects within these categories.

Rocket was the most difficult category for all three models. DGCNN achieved 55.1 ± 11.7%, Point Transformer 58.2 ± 11.2%, and PointNet++ 60.9 ± 10.4%, making PointNet++ the strongest model on this category despite its lower aggregate score. In contrast, Laptop was segmented reliably by every architecture, with mIoU between 95.1% and 95.9%. DGCNN produced the best Laptop result and also led on Airplane, Guitar, Knife, and Lamp. PointNet++ led on Pistol, Rocket, and Skateboard. Point Transformer performed best on Bag, Cap, Car, Chair, Earphone, Motorbike, Mug, and Table.

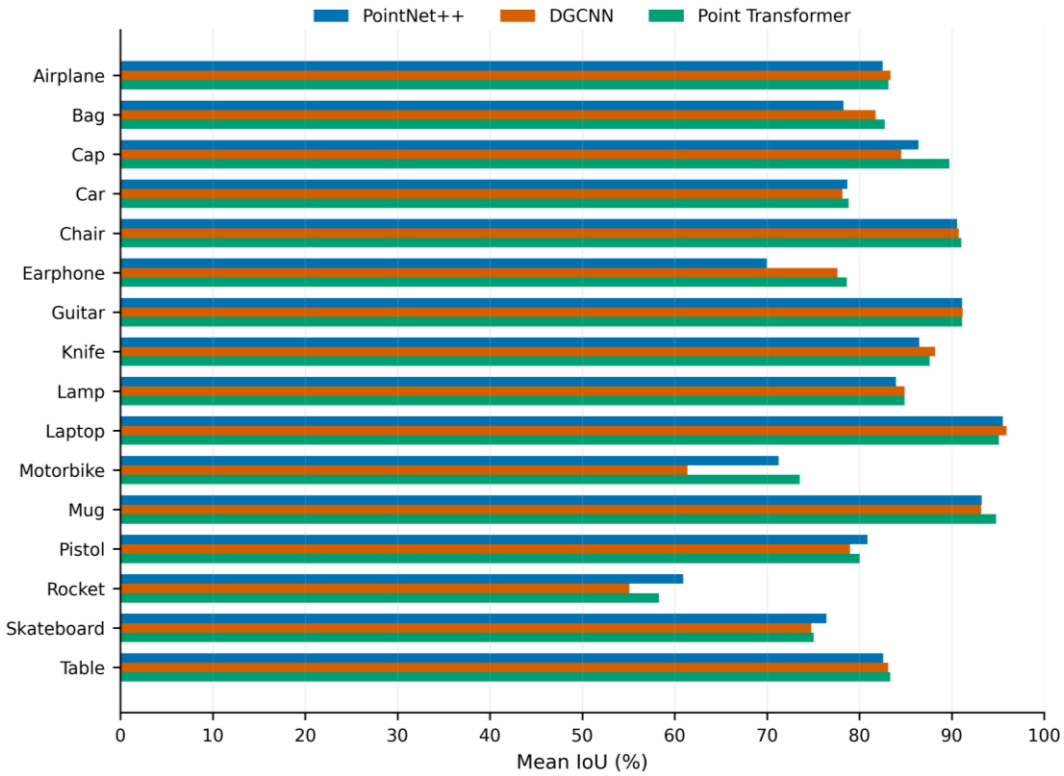


Fig. 3. Per-category mIoU comparison for PointNet++, DGCNN, and Point Transformer on ShapeNet Part

Overall, the per-category comparison demonstrates that aggregate mIoU should be supplemented by category-level analysis and measures of variability across individual objects when segmentation reliability for a specific object family matters.

Computational Efficiency and Quality–Efficiency Trade-off

The three networks differed much more strongly in inference efficiency than in segmentation quality. **Table 3** summarizes the batch-size-one benchmark. DGCNN was the fastest model, requiring 15.87 ms per shape and processing approximately 62.99 shapes/s. Point Transformer required 53.39 ms per shape and reached 18.73 shapes/s. PointNet++ was substantially slower in the evaluated implementation, with a mean latency of 344.62 ms per shape and throughput of 2.90 shapes/s.

Table 3. **Measured computational efficiency in the batch-size-one benchmark**

Model	Latency ms	Rate shapes/s	GPU MB
PointNet++	344.62	2.90	132.9
DGCNN	15.87	62.99	157.7
Point Transformer	53.39	18.73	99.1

Point Transformer showed the lowest maximum allocated graphics memory during the measured batch-size-one inference, requiring 99.1 MB, compared with 132.9 MB for PointNet++ and 157.7 MB for DGCNN.

Fig. 4 makes the practical trade-off especially clear. Moving from DGCNN to Point Transformer increases instance mIoU by approximately 0.4 percentage points, but increases mean latency by a factor of about 3.4. PointNet++ does not provide an aggregate

quality advantage over the other two models in this benchmark and is approximately 6.5 times slower than Point Transformer and 21.7 times slower than DGCNN. These values characterize the tested implementations and should not be interpreted as architecture-independent theoretical speed ratios; nevertheless, they are directly relevant to deployment when the same software and hardware stack is used.

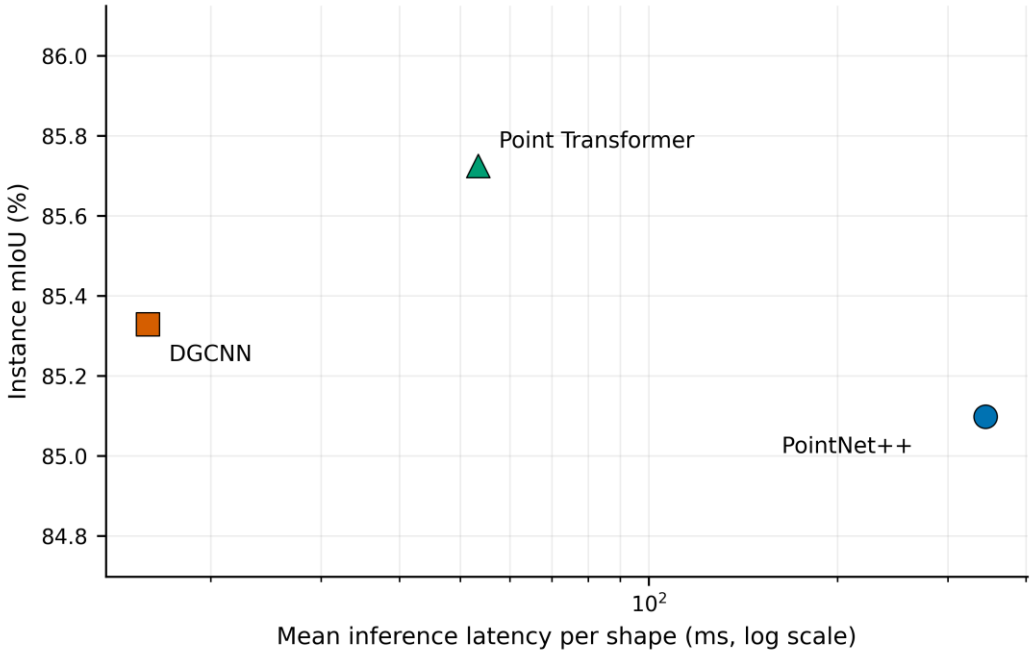


Fig. 4. Quality–efficiency trade-off between instance mIoU and mean inference latency. The latency axis is logarithmic.

Practical Trade-offs and Study Limitations

The experimental results demonstrate that model selection depends on the intended application. Point Transformer is preferable when the primary objective is maximum segmentation quality and a moderate increase in inference cost is acceptable. DGCNN provides the strongest measured speed-quality compromise: its instance mIoU is only 0.40 percentage points below Point Transformer while its inference is substantially faster. PointNet++ remains a competitive segmentation baseline and performs best on several categories, including Rocket, but its measured latency makes it less attractive for low-latency deployment in the evaluated implementation.

Several limitations should be considered when interpreting the results. First, the experiments use a single synthetic/CAD-derived benchmark and therefore do not quantify robustness to sensor noise, occlusion, missing points, or non-uniform sampling in real-world scans. Second, only three architectures were evaluated, so the results do not represent the full range of recent point-cloud segmentation methods. Third, all efficiency measurements were obtained on one hardware platform and are influenced by repository-specific kernels and implementation details.

Finally, ShapeNet Part evaluates object-level part segmentation rather than semantic segmentation of large scenes. The conclusions are therefore most directly applicable to normalized object point clouds. Extending the comparison to real sensor data, larger scenes, and downstream geometric measurements will be necessary to determine how the observed trade-offs transfer to practical three-dimensional perception systems.

CONCLUSION

This study compared PointNet++, DGCNN, and Point Transformer for part segmentation of point clouds on the ShapeNet Part benchmark. The final evaluation used 2,874 test shapes, 2,048 points per object, and a common batch-size-one inference protocol. Segmentation performance was analyzed using point accuracy, class-average accuracy, class mIoU, instance mIoU, per-category mIoU, training dynamics, and computational efficiency.

Point Transformer achieved the highest aggregate segmentation quality, reaching 94.4% point accuracy, 82.9% class mIoU, and 85.7% instance mIoU. DGCNN and PointNet++ remained close in overall quality, with instance mIoU values of 85.3% and 85.1%, respectively. The small aggregate differences, however, concealed substantially larger category-specific variation. Point Transformer produced the highest mIoU in eight categories, DGCNN in five, and PointNet++ in three. Motorbike, Earphone, and Rocket showed the most informative architecture-dependent differences, while Laptop was consistently segmented with high accuracy by all models.

The computational benchmark revealed a much stronger separation between the architectures. DGCNN achieved the lowest mean latency of 15.87 ms per shape and the highest throughput of 62.99 shapes/s. Point Transformer required 53.39 ms per shape, whereas PointNet++ required 344.62 ms. These results show that a model with the highest segmentation quality is not automatically the most suitable model for a latency-constrained system.

Overall, Point Transformer provides the strongest segmentation quality in the evaluated setup, while DGCNN offers the most favorable measured speed-quality trade-off. PointNet++ remains useful as a hierarchical baseline and demonstrates strengths on selected categories, but its inference cost was substantially higher in the tested implementation. The results confirm that architecture selection should combine aggregate mIoU with category-level reliability and deployment-oriented efficiency metrics.

Future work will focus on methods for determining geometric and physical object parameters from three-dimensional point cloud data. The study will consider point clouds containing isolated objects of different simple geometric shapes and will investigate both algorithmic and neural-network-based approaches. Classification models will be used to identify the geometric type of an object, after which geometry-specific algorithms will estimate its relevant dimensions, shape characteristics, and derived physical parameters. The proposed framework will enable a systematic comparison of analytical and learning-based approaches and will investigate whether combining object recognition with geometry-aware parameter estimation can improve the reliability and general applicability of measurements obtained directly from point clouds.

ACKNOWLEDGMENTS AND FUNDING SOURCES

The author received no financial support for the research, writing, and publication of this article.

COMPLIANCE WITH ETHICAL STANDARDS

The authors declare that the research was conducted in the absence of any conflict of interest.

AUTHOR CONTRIBUTIONS

Conceptualization, [V.V., Y.F.]; methodology, [V.V.]; validation, [V.V.; formal analysis, [V.V.]; investigation, [V.V.]; resources, [V.V.]; data curation, [V.V.]; writing – original draft

preparation, [V.V.]; writing – review and editing, [V.V.]; visualization, [V.V.]; supervision, [V.V., Y.F.]; project administration, [V.V., Y.F.].

All authors have read and agreed to the published version of the manuscript.

DECLARATION ON GENERATIVE AI

During the preparation of this manuscript, the authors used ChatGPT for language editing, improving the structure and flow of author-provided text, and assistance with formatting according to the journal template. The tool was not used to generate experimental data, numerical benchmark results, citations, or scientific conclusions. After using ChatGPT, the authors reviewed and edited the resulting text and take full responsibility for the content of the publication.

REFERENCES

[1] Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., & Bennamoun, M. (2021). Deep learning for 3D point clouds: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 43(12), 4338–4364. https://doi.org/10.1109/TPAMI.2020.3005434

[2] Qi, C. R., Su, H., Mo, K., & 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, 652–660. https://doi.org/10.1109/CVPR.2017.16

[3] Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017). PointNet++: Deep hierarchical feature learning on point sets in a metric space. Advances in Neural Information Processing Systems, 30. https://doi.org/10.48550/arXiv.1706.02413

[4] Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., & Solomon, J. M. (2019). Dynamic graph CNN for learning on point clouds. ACM Transactions on Graphics, 38(5), 1–12. https://doi.org/10.1145/3326362

[5] Zhao, H., Jiang, L., Jia, J., Torr, P. H. S., & Koltun, V. (2021). Point Transformer. Proceedings of the IEEE/CVF International Conference on Computer Vision, 16259–16268. https://doi.org/10.1109/ICCV48922.2021.01595

[6] Yi, L., Kim, V. G., Ceylan, D., Shen, I.-C., Yan, M., Su, H., Lu, C., Huang, Q., Sheffer, A., & Guibas, L. (2016). A scalable active framework for region annotation in 3D shape collections. ACM Transactions on Graphics, 35(6), 1–12. https://doi.org/10.1145/2980179.2980238

[7] Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., & Yu, F. (2015). ShapeNet: An information-rich 3D model repository. arXiv. https://doi.org/10.48550/arXiv.1512.03012

[8] Wu, X., Jiang, L., Wang, P.-S., Liu, Z., Liu, X., Qiao, Y., Ouyang, W., He, T., & Zhao, H. (2024). Point Transformer V3: Simpler, faster, stronger. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 4840–4851. https://doi.org/10.1109/CVPR52733.2024.00463

[9] Ando, A., Gidaris, S., Bursuc, A., Puy, G., Boulch, A., & Marlet, R. (2023). RangeViT: Towards vision transformers for 3D semantic segmentation in autonomous driving. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 5240–5250. https://doi.org/10.1109/CVPR52729.2023.00507

[10] Zhang, J., Luo, Y., Zhang, Z., Nie, X., & Li, B. (2025). CamPoint: Boosting point cloud segmentation with virtual camera. Proceedings of the IEEE/CVF Conference

on Computer Vision and Pattern Recognition, 11822–11832. https://doi.org/10.1109/CVPR52734.2025.01104
[11] Yan, X. (2019). *Pointnet_Pointnet2_pytorch: PyTorch implementation of PointNet and PointNet++* [Source code]. GitHub. https://github.com/yanx27/Pointnet_Pointnet2_pytorch
[12] Tao, A. (n.d.). *dgcnn.pytorch: A PyTorch implementation of Dynamic Graph CNN for Learning on Point Clouds* [Source code]. GitHub. https://github.com/antao97/dgcnn.pytorch
[13] Ji, J. (n.d.). *Point-Transformer: A PyTorch implementation of Point Transformer for batch-mode point cloud processing* [Source code]. GitHub. https://github.com/ji-code25/Point-Transformer
[14] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., & Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. Advances in Neural Information Processing Systems, 32. https://doi.org/10.48550/arXiv.1912.01703

СЕГМЕНТАЦІЯ СКЛАДНИХ СТРУКТУР У ХМАРАХ ТОЧОК ЗА ДОПОМОГОЮ ГЛИБИННИХ НЕЙРОННИХ МЕРЕЖ

Віктор Вдовиченко*, Юрій Фургала
Факультет електроніки та комп'ютерних технологій,
Львівський національний університет імені Івана Франка,
вул. Драгоманова 50, Львів, 79005, УКРАЇНА

АНОТАЦІЯ

Вступ. Сегментація частин хмар точок призначає семантичну мітку кожній точці та забезпечує детальніший опис геометрії об'єкта, ніж класифікація цілого об'єкта. Вибір архітектури ускладнюється тим, що моделі з близькими інтегральними показниками можуть істотно відрізнятися в окремих категоріях і за обчислювальною ефективністю.

Матеріали та методи. Для порівняльного оцінювання обрано три репрезентативні архітектури обробки хмар точок, що відображають різні підходи до вилучення геометричних ознак: PointNet++ (ієрархічна агрегація локальних ознак), DGCNN (динамічна побудова графа сусідів і графові згортки) та Point Transformer (механізм самоуваги для неструктурованих даних). Порівняння проведено на наборі ShapeNet Part (16 категорій об'єктів, 50 міток частин).Фінальне оцінювання виконано на 2874 тестових об'єктах, представлених 2048 точками. Якість визначали за точністю точок, середньою точністю за класами, середнім коефіцієнтом Жаккара (КЖ) за категоріями та середнім КЖ за об’єктами. Для вимірювання затримки виведення, пропускної здатності та обсягу використаної відеопам'яті було проведено окремий тест із розміром пакета 1.

Результати. Point Transformer показав найкращу загальну якість: точність 94,4%, середня точність за класами 87,7%, середній КЖ за категоріями 82,9% та середній КЖ за об'єктами 85,7%. При цьому DGCNN досягла близького показника середнього КЖ за об'єктами (85,3%), але значно швидше: 15,87 мс на один об'єкт проти 53,39 мс у Point Transformer і 344,62 мс у PointNet++. Загальні відмінності між моделями були невеликими, проте результати за окремими категоріями відрізнялися суттєвіше: Point

Transformer мав найвищий середній КЖ у восьми з шістнадцяти категорій, DGCNN – у п'яти, а PointNet++ – у трьох. Найбільший розкид значень КЖ між моделями спостерігався для категорій "мотоцикл" (Motorbike), "навушники" (Earphone) і "ракета" (Rocket) – саме на них вибір архітектури мав найбільше значення, тоді як категорію "ноутбук" (Laptop) усі моделі сегментували стабільно точно.

Висновки. Жодна архітектура не була найкращою за всіма критеріями. Point Transformer забезпечив найвищу якість сегментації, тоді як DGCNN показала найкраще виміряне співвідношення швидкості та якості. PointNet++ залишилася конкурентною за якістю, але мала значно більшу затримку в оціненій реалізації. Вибір моделі має враховувати поведінку на окремих категоріях, конфігурацію вхідних ознак та обчислювальні обмеження разом із загальними значеннями середнього КЖ.

Ключові слова: сегментація хмар точок, сегментація частин, ShapeNet Part, PointNet++, DGCNN, Point Transformer

Received / Одержано	Revised / Доопрацьовано	Accepted / Прийнято	Published / Опубліковано
02 September, 2026	20 September, 2026	22 September, 2026	28 September, 2026