Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 44 No. 5 (2023)

Comprehensive Study and Analysis of Point Transformer for Point Cloud Data

[1]Ms. Harita Parikh, [2]Dr. Purvi Prajapati, [3]Dr. Nirav Bhatt, [4]Dr. Nikita Bhatt, [5]Ms. Jahnvi Shah

[1]Ms. Harita Parikh, B.Tech IT student, Smt. K. D. Patel Department of Information Technology, CSPIT, Charotar University of Science and Technology, Changa, Gujarat, India
[2]Dr. Purvi Prajapati, Assistant Professor, Smt. K. D. Patel Department of Information Technology, CSPIT, Charotar University of Science and Technology, Changa, Gujarat, India.
[3]Dr. Nirav Bhatt, Associate Professor, Department of Artificial Intelligence and Machine Learning, CSPIT, Charotar University of Science and Technology, Changa, Gujarat, India .
[4]Dr. Nikita Bhatt, Assistant Professor, U & P U. Patel Department of Computer Engineering, CSPIT, Charotar University of Science and Technology, Changa, Gujarat, India.
[5]Ms. Jahnvi Shah, B.Tech IT student, Smt. K. D. Patel Department of Information Technology, CSPIT, Charotar University of Science and Technology, Changa, Gujarat, India.

Abstract: Self-attention networks are making significant progress in picture analysis tasks like image categorization and object detection, just as they have revolutionized natural language processing. This paper represents the use of self-attention networks for processing point cloud data in response to this achievement. To build self-attention networks for tasks like object categorization, this paper has explored self-attention layers for point clouds. It is used in many applications such as Point Cloud Classification, Object Detection, Point Cloud Generation etc. This research paper covers the basics of Point Transformer and its importance in Point Cloud Data. the performance of the Point Transformer algorithm was evaluated on benchmark datasets, such as ModelNet10 with 92.40% accuracy.
Keywords: Deep Learning, Neural Network, Machine Learning, Recommendation

1. INTRODUCTION

The data type known as point clouds represents regions or objects in space by using the X, Y, and Z coordinates of individual points on a sampled surface. This collection of data points can be used to create a comprehensive dataset of an entire scene. When colour information is also added, then the point cloud becomes four-dimensional. To generate point clouds, the most commonly used technologies are 3D laser scanners and LiDAR (Light Detection and Ranging) [18, 57]. LiDAR works by using a laser to measure the distance of a surface from a scene. Multiple laser beams are fired and reflected from objects to create a set of points that depict the scene's geometry.

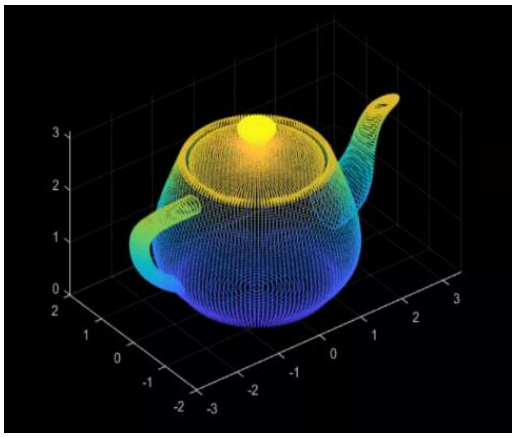

Figure 1. Point cloud representation of kettle

The use of transformers for image analysis and natural language processing act as inspiration for the

study's unique approach to handling point clouds with deep learning [56]. The self-attention operator of the transformer network, which is unaffected by the order or number of input elements, is useful for point cloud processing. This research focuses on developing a self-attention layer specifically for processing 3D point clouds and using it to construct Point Transformer networks for various 3D comprehension tasks [4]. Because 3D point clouds are a significant component of 3D space. The resulting networks rely only on pointwise operations and self-attention. Table 1 provides examples of completed projects using the Point Transformer.

Table 1: Applications areas for "Point Transformer" domain

<table>
<tr><th>Application Domain</th><th>Overview</th></tr>
<tr><td>Point Cloud Classification</td><td>The process of giving each point in a point cloud is a class designation [57]. In a number of benchmark datasets, including ModelNet40 and ScanObjectNN, The Point Transformer has produced cutting-edge results.</td></tr>
<tr><td>Point cloud segmentation</td><td>The task of dividing a point cloud into semantic regions, with each point belonging to a different region. On datasets like ShapeNetPart and S3DIS, the Point Transformer has shown promising results.</td></tr>
<tr><td>Object Detection</td><td>This domain is used for locating and identifying things in a point cloud. In datasets like KITTI and SemanticKITTI, the Point Transformer has been used for this task and produced results that are competitive.
<table>
<tr><th>Sensor</th><th>x-axis direction</th><th>y-axis direction</th><th>z-axis direction</th></tr>
<tr><td>Camera</td><td>Right</td><td>Down</td><td>Forward</td></tr>
<tr><td>LiDAR</td><td>Forward</td><td>Left</td><td>Up</td></tr>
<tr><td>GPS/IMU</td><td>Forward</td><td>Left</td><td>Up</td></tr>
</table></td></tr>
<tr><td>Point Cloud Generation [59]</td><td>The task of generating new point clouds that are similar to a given input point cloud. The Point Transformer has been used in this task to generate high-quality point clouds on the ModelNet40 dataset [52, 56, 59].</td></tr>
</table>

The remaining part of the paper is organized as follows. The existing work in Point cloud is described in Section 2. Section 3 discussed Graph Neural Network followed by Transformer and Point Transformer details in section 4 and 5 respectively. Section 6 represented proposed design approach. Last two sections cover the experimental discussion and conclusion.

2. Related Work

Although pixels in 2D images are structured and conventional convolution works well to analyze them, but 3D point clouds are haphazardly strewn throughout 3D space, they are more difficult to analyze. The article then divides learning-based methods for dealing with 3D point clouds into three groups: (i) projection-based networks, (ii) voxel-based networks, and (iii) point-based networks.

In order to process irregular inputs like point clouds, projection-based networks turn their irregular representations into regular ones. Multi-view projection, which projects 3D point clouds onto different picture planes, is used to achieve this [4]. Then, feature representations are extracted from these surfaces using 2D CNNs [56]. In contrast to 2D CNNs, which are efficient at processing routine inputs, this is different. The output representations are produced by multi-view feature fusion. Another method that is comparable to projection-based networks is TangentConv. Local surface geometry is projected onto a tangent at each location, resulting in the creation of a tangent. Yet, tangent estimation has a significant impact on TangentConv's correctness. One limitation of using projection in these frameworks is that the geometric data contained in point clouds may be lost during the projection stage. Furthermore, these methods might not utilize the points' spatial knowledge to

their maximum potential when creating dense pixel grids on projection planes [56]. This can result in a decrease in accuracy, especially when dealing with occlusion.

By using 3D voxelization and 3D convolutions, voxel-based networks offer an alternate technique for transforming irregular point clouds into regular representations. However, due to the cubic growth in voxel count as the resolution grows, if the number of voxels is not controlled effectively, the computational and memory expenses might be enormous. These techniques make use of sparsity because the majority of voxels are usually vacant. OctNet combines imbalanced octrees with hierarchical partitions to lower the computational and memory needs. Sparse convolution techniques further reduce the amount of processing and memory required by only evaluating the convolution kernel at occupied voxels. Despite the accuracy of these methods, the quantization of the voxel grid may cause some geometric detail to be lost.

Researchers have built point-based networks, in contrast to projecting or quantizing point clouds onto traditional grids in 2D or 3D, direct analysis of point clouds as sets imbedded in continuous space is preferred [56]. PointNet uses permutation-invariant operators like pointwise MLPs and pooling layers to integrate features across a collection [56]. PointNet++ improves sensitivity to the local geometric arrangement inside a hierarchical spatial framework by utilizing these concepts. Many sampling techniques are used by some models, including [7, 11, 27, 46, 50], to effectively sample the point set.

To create connections between point sets and conduct message passing on the resulting graph, various strategies have been presented. Graph convolutions in DGCNN are performed on kNN graphs, but in PointWeb, dense connections are made between nearby neighborhoods. Many techniques apply continuous convolutions without quantization on the 3D point set directly. PCCN represents convolutional kernels using MLPs [42]. Kernel weights are described by SpiderCNN as a family of polynomial functions [49]. Spherical in order to overcome the problem of 3D rotation equivariance, CNN uses spherical convolution [8]. Based on the input coordinates, PointConv and KPConv compute convolution weights [37,46]. Using coordinates, InterpCNN interpolates pointwise kernel weights [22]. In order to reorganize the input unordered point clouds, PointCNN advises utilizing special operators [2, 20]. Ummenho and colleagues apply continuous convolutions to learn particle-based fluid dynamics [38, 56, 58].

3. GRAPH NEURAL NETWORK

Graph Neural Networks (GNN) is a modern, data-driven approach to analyzing large graphs and discovering their structure. A GNN consists of three key components: (i) information associated with each node (and optionally each edge) in the graph, represented as a single-dimensional vector known as an embedding, (ii) a recursive message-passing aggregation algorithm that aggregates vertex/edge embedding in novel ways and (iii) a neural network model (typically an MLP), that learns the graph structure iteratively via standard gradient-descent training algorithms used in DL. The impact of GNNs on the modern world cannot be overstated – they have the power to analyze complex relationships at both the macro scale such as the Internet or social media networks and the micro-scale of protein molecules in the human and their interactions with drug molecules to aid in drug discovery. Conducting such analyses on graphs via machine learning techniques requires specifying tasks such as node classification, link prediction, graph classification, community detection, and network similarity. The GNN approach can augment, rather than substitute, traditional graph analytics to achieve the "best-of-both-worlds" – accuracy of exact graph analysis combined with the speed of training neural networks.

There are two broad flavours of GNN training – full-batch training and mini-batch training via sampling techniques. Depending on the type of graph data, a data scientist may choose either flavor for their application. In full-batch training, all nodes in the training dataset participate in the computation in every iteration; because each vertex is associated with a dense feature vector, full-batch training on large graphs requires a large amount of memory to store both the embedding and their gradients. On the other hand, mini-batch training selects a set of vertices and associated sampled neighborhoods from the training dataset every iteration; if the vertex set size is small and the extent of the sampled neighborhood around each vertex is limited, then mini-batch training presents lower memory capacity requirements. In either flavor, the two key primitives involved in the computation are (i) Aggregation and (ii) Update. Figure 2 pictorially represents the entire process of GNN training using these two primitives. We now describe these two primitives.

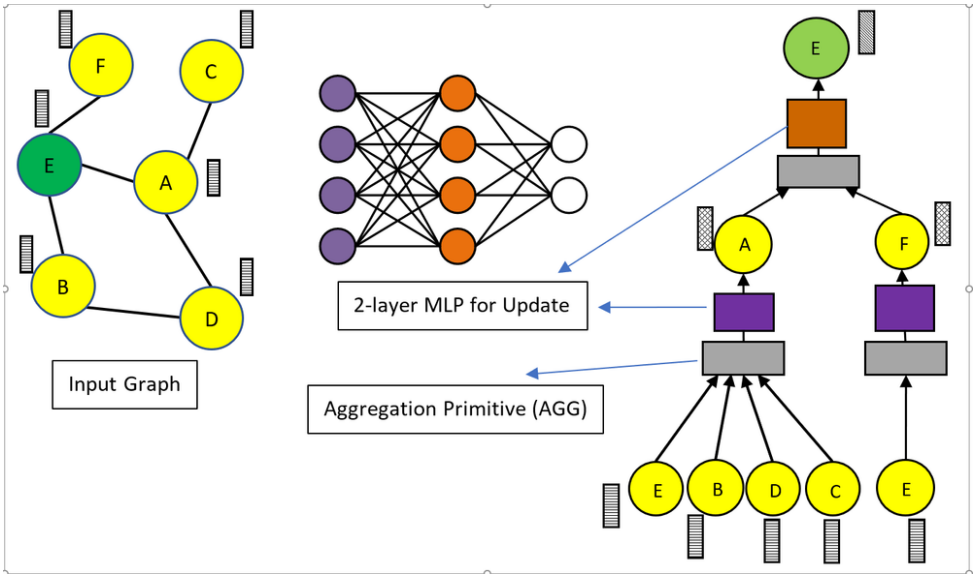

Figure 2. Aggregation and Update in GNN Training for vertex

3.1 Aggregation Primitive

Let G (V, E) be the input graph with vertices V and edges E and let Fv and Fe be the associated vertex and edge features, respectively. The sizes of Fv and Fe are |V| x dand |E| x d respectively, where d is the feature-vector size. We can represent Aggregation as a tuple (Fv, Fe, $\otimes$, $\oplus$, Fo), where $\otimes$ and $\oplus$ are element-wise operators on Fv or Fe (or a combination thereof) and Fo is the output feature associated with a vertex or edge. $\otimes$ can be an element-wise binary or unary operator. In binary form, it operates on a pair of inputs; valid pairs are (Fv, Fv) and (Fv, Fe), in an appropriate order. The operator $\oplus$ performs element-wise reduction on the result of the binary operation to the final output. Mathematically, if AGG represents the Aggregation operation, then

$$\mathrm{AGG}(x, y, \otimes, \oplus, z) : \oplus\,(\otimes\,(x, y), z) \forall x, y, z \in Fv, Fe$$

If one of the operands, say y does not exist, then $\otimes$ becomes a unary operator: it reduces each instance of operand x (using the reduction operator) into the final output feature-vector z. In Figure 2, the grey-colored boxes in each layer represent AGG. Going from bottom to top, AGG aggregates features associated with vertices B, C, D, and E, respectively, in the first layer and those associated with A and F, respectively in the second layer. We discuss the performance implications of the Aggregation operator in some detail later in this article.

3.2 Update Primitive

The Update primitive is typically an MLP consisting of a Linear operator (e.g., torch.nn.Linear) followed by one or more element-wise operators (e.g., Rectified Linear Unit (ReLU), DropOut, etc.). The Linear operator contains learnable parameters (i.e., weights and biases) that together constitute the GNN model. As shown in Figure 1, the model weights are represented as connections between the purple- and brown-colored neurons, as well as between the brown- and white-colored neurons in the example 2-layer neural network. Thus, the Update primitive produces an updated set of vertex embedding by filtering them through model parameters after the Aggregation primitive aggregates initial vertex embedding.

4. TRANSFORMER

By examining interactions between various data items, the Transformer model, a sort of neural network, excels at comprehending the context and meaning of sequential data, such as sentences. It employs attention, or self-attention, which makes use of mathematical methods to ascertain how various pieces in a series connect to one another, even if they are spread out over a large distance. The encoder and decoder are the two components that make up the Transformer, which is intended to change one sequence into another. The Transformer does not make use of recurrent networks, in contrast to earlier sequence-to-sequence models (GRU,

LSTM, etc.).

The Transformer model's self-attention process involves determining attention weights for each element in the input sequence depending on how those elements relate to one another. The next layer of the network receives the weighted sum of the input sequence that was computed using these attention weights. This enables the network to process the input sequence in parallel by allowing it to choose focus on various portions of the sequence. This varies from standard neural network topologies like recurrent neural networks or convolutional neural networks, where each element in the sequence is processed one at a time. The self-attention method is shown in Figure 3 with Q standing for the query, K for the key, and V for the values.

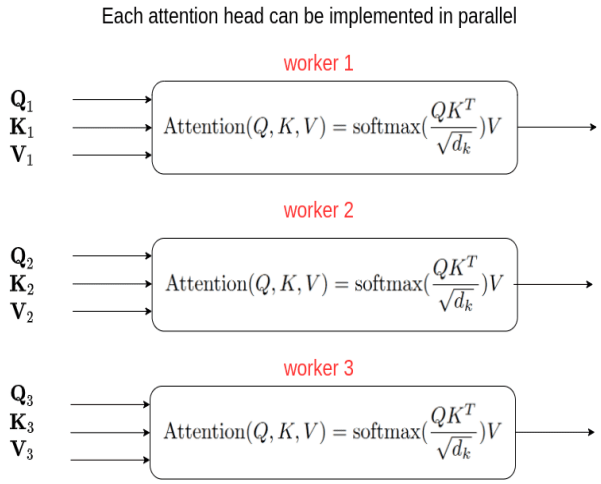

Figure 3. Working of self-attention

For instance, the search engine will compare your query to a set of keys (such as video titles and descriptions) connected with probable videos in the database if you input a search query on YouTube to discover a video. The best videos (values) that match your search will then be shown. Multi-head attention and feedforward neural networks are two crucial elements of the Transformer design. While feedforward neural networks offer non-linear changes of the input sequence, multi-head attention enables the network to concentrate on different points of the input sequence at once. Layer normalization and residual connections, which aid in training stability and performance enhancement, are also included in the architecture.

The model in question is a recurrence-free variation of recurrent neural networks (RNN). Before passing through multi-head attention, positional information must be supplied to the input embedding's to account for the lack of repetition. The given equation was utilized for this positional encoding.

$$p_{ij} = \begin{cases} \sin\left(\frac{i}{10000^{\frac{j}{d}}}\right) \\ \cos\left(\frac{i}{10000^{\frac{j}{d}}}\right) \end{cases} \qquad \text{(Eq. 1)}$$

If an object's position in a given input sequence is even, the sine function is applied, and if it is odd, the cosine function is applied. A huge value of 10,000 is used to ensure that the complete cycle is exceedingly large, which is required for encoding words, which could be longer than 1,000 words. The value of j is mapped to both sine and cosine using the map column indices. The dimension of the output embedding space is d.

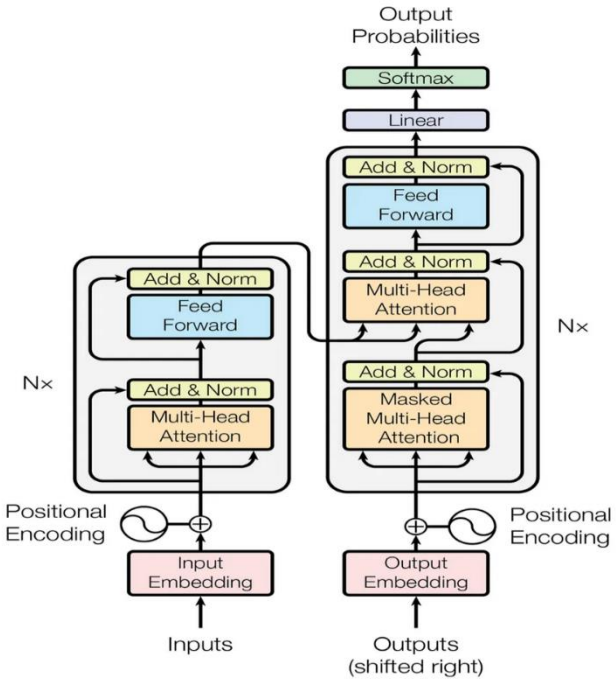

Figure 4. Encoder and Decoder structure of a transformer

The Encoder is shown in Figure 4 on the left side, while the Decoder is shown in Figure 4 on the right side. The "Nx" notation indicates that encoder-decoder modules that can be layered many times. Feed Forward and Multi-Head Attention layers make up the majority of these modules. The input and output sentences are first turned into an n-dimensional space through embedding because text strings cannot be processed directly.

5. POINT TRANSFORMER

The Point Transformer is a neural network architecture designed for point cloud processing, which is the task of processing data represented in 3D space as a set of points. The Point Transformer is based on the Transformer architecture, which was originally designed for NLP tasks, and it aggregates information from neighboring points using a self-attention mechanism [58]. Point Transformer represents each point in the point cloud as a feature vector. These feature vectors are then processed by the Point Transformer's multiple layers, each of which consists of a self-attention module followed by a feedforward neural network. The self-attention module enables the network to aggregate information from neighboring points, whereas the feedforward network performs non-linear transformations on the feature vectors.

The use of relative positional encoding is one of the Point Transformer's key innovations. Relative positional encoding is based on the relative positions of points within a local neighborhood, and it is used to encode spatial relationships [58]. This method applies convolutional filters to a point cloud, allowing the network to capture local spatial relationships between points. The dynamic set convolutions are used in the Point Transformer's feedforward network shown in Figure 5. They are computed based on the relative positions of points within a neighborhood.

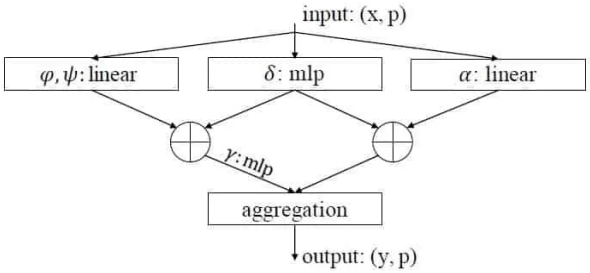

Figure. 5 The point transformer layer

Given that the attention is calculated over each point's k nearest neighbors, the resulting Point Transformer layer is localized. MLP with two linear layers and a ReLU nonlinearity is a mapping function that is utilized to calculate vector attention. Both the attention and the value vectors are added, and the latter two are then multiplied to produce the output. Trainable positional encodings are computed using a separate MLP with two linear layers and a ReLU. The Point Transformer network for segmentation tasks is structured as a U-Net, with 5 down sampling point attention layers, 5 up sampling layers, and at last MLP layer, with skip connections between the down and up layers of the same output size. Only the down sampling part is used at the end for classification tasks, with an average pooling layer and MLP layer. The Table 2 discusses the comparative study of Point Transformer, Projection Based Networks, Voxel Based Networks, and, Point Based Networks.

Table 2: Comparative study

POINT TRANSFORMER A recently created method for analyzing point clouds called Point Transformer is based on the Transformer architecture. In order to learn feature representations for specific points in the point cloud, it employs self-attention techniques. Message passing is then used to aggregate features across nearby neighborhoods. In the classification, segmentation, and object recognition of point clouds, Point Transformer has demonstrated encouraging results [58].
PROJECTION BASED NETWORKS Point cloud data is projected onto a 2D or 3D grid by projection-based networks, also known as image-based networks. Standard convolutional neural network (CNN) procedures are then used to extract features from the projected data. This method, where the point cloud data is projected onto a grid for a bird's eye perspective, is frequently employed in lidar-based perception for autonomous driving.
VOXEL BASED NETWORKS The point cloud data is discretized into a 3D voxel grid by voxel-based networks, which then use CNN procedures to extract characteristics from the voxel data. This method discretizes the 3D space into a grid, which is like projection-based networks in that it does not project the point cloud onto a 2D grid. For object detection and point cloud segmentation, voxel-based networks are frequently used.
POINT BASED NETWORKS Point-based networks, sometimes referred to as point-based convolutional neural networks (PointCNN), work directly with the unprocessed point cloud data, without projection or discretization. Point-based networks learn features from nearby neighborhoods of points using techniques like ball query and farthest point sampling, then aggregate the features over various scales. In the classification, segmentation, and registration of point clouds, point-based networks have demonstrated promising results.

6. PROPOSED DESIGN

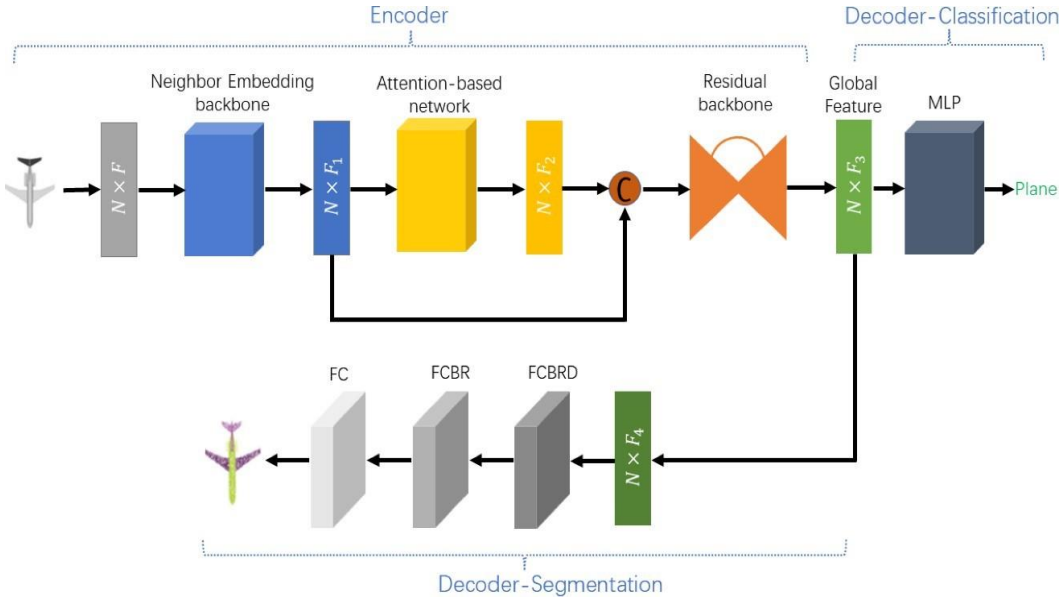

Figure. 6 Proposed Design

Figure 6 depicts the Point Transformer's overall design, which includes decoders for various tasks, a neighborhood embedding backbone, an attention-based sub-network, and a residual backbone. Transformer, which initially encodes the input characteristics into a new high-dimensional feature space. For many point cloud processing tasks, the semantic associations between points are expressed in this way. To learn about the nearby localities, it first embeds the input coordinates of the point cloud into a new space. The four stacked offset-attention layers that make up the attention-based sub-network enable it to develop semantically rich and discriminatory representations for each point. Then, to utilize the context information of the point cloud, we take the output feature of the attention-based sub-network into the residual backbone.

During the categorization exercise, the global feature is input into the classification decoder, which has MLP layers (1024, 512, 256, and Nc), and dropout operation with an invariable probability of 0.5 to convert the global feature to Nc object categories, to identify Nc object categories in point cloud P. Additionally, we employ batch normalization together with the activation function LeakyReLU in each layer. Similar criteria are used when choosing other hyperparameters. The category label for this point cloud is chosen based on the highest-scoring category. In the task of portion segmentation. Obtaining the precise semantic label for each point is necessary if we are to segment the point cloud into Ns parts (such as cub handles and airplane wings; a component hardly requests to be contiguous). As shown in Figure 1, the part segmentation decoder uses three shared full-connected layers (512, 256, and Ns) to categorize each point using the global feature produced by the residual backbone. To be more specific, the activation function ReLU and a dropout layer with a probability of 0.5 are placed after the first fully connected layer. On the second full-connected layer, only the activation function ReLU is used. All layers are additionally batch-normalized.

7. EXPERIMENTAL DISCUSSION

ModelNet10 dataset is a part of ModelNet40 dataset, containing 4,899 pre-aligned shapes from 10 categories. There are 3,991 (80%) shapes for training and 908 (20%) shapes for testing. The CAD models are in Object File Format (OFF). MATLAB functions to read and visualize OFF files are provided in Princeton Vision Toolkit (PVT). To build the core of the dataset, a list of the most common object categories in the world was compiled, using the statistics obtained from the SUN database. Once a vocabulary for objects was established, 3D CAD models belonging to each object category was collected using online search engines by querying for each object category term. Then, human workers on Amazon Mechanical Turk were hired to manually decide whether each CAD model belonged to the specified categories, using an in-house designed tool with quality control. To obtain a very clean dataset, 10 popular object categories were chosen while manually deleted the models that did not belong to these categories. Furthermore, manual alignment of orientation of CAD models

was performed for the 10-class subset.

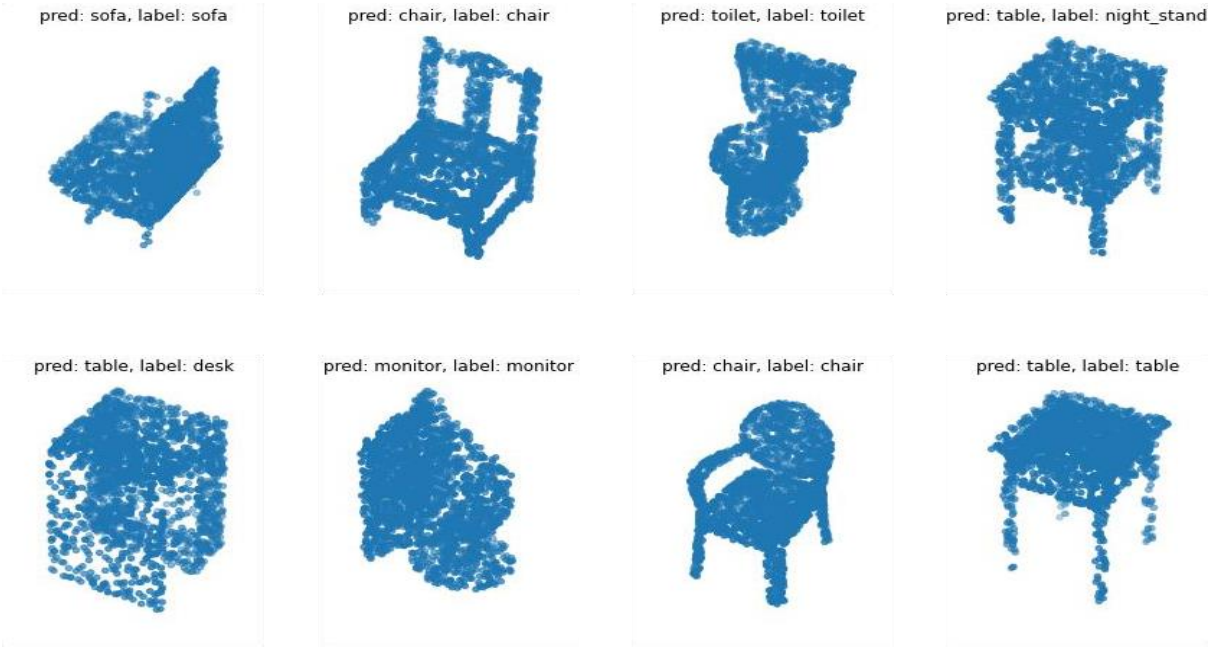

Figure. 7 Modelent10 Point Cloud Image

Evaluate the performance of the Point Transformer algorithm on benchmark datasets, such as ModelNet10. This includes measuring accuracy, speed, memory usage, and scalability.

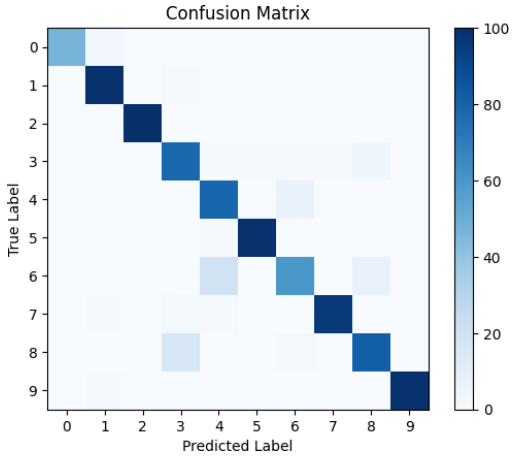

Figure. 8 Confusion Matrix of ModelNet10 dataset

A confusion matrix is used in machine learning to evaluate the performance of a classification model. It is a matrix of actual vs predicted values, and it helps to visualize the true positives, true negatives, false positives, and false negatives. As per the Fig. 8, in confusion matrix almost true positives for all the class lies between 50-100. The accuracy of the model is 92.40% and loss of the model is 3.46%.

8. Conclusion

The Point Transformer has demonstrated cutting-edge performance on a wide range of NLP (Natural Language Processing) tasks, including machine translation, language modeling, and text categorization. It has also been effectively used in other fields, such as speech recognition, music generation, image processing and video processing. The Transformer's success has led to the development of various extensions and adaptations, such as the BERT (Bidirectional Encoder Representations from Transformers) model and the GPT (Generative Pre-Trained Transformer) series. Its impact on the field of deep learning has been significant, and it has become

a critical component of many cutting-edge neural network architectures.

REFERENCES

[1] Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3D semantic parsing of large-scale indoor spaces. In CVPR, 2016.
[2] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In CVPR, 2017.
[3] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In CVPR, 2019.
[4] Zhao, Hengshuang, et al. "Point transformer." Proceedings of the IEEE/CVF international conference on computer vision. 2021.
[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In NAACL-HLT, 2019.
[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. ICLR, 2021.
[7] Oren Dovrat, Itai Lang, and Shai Avidan. Learning to sample. In CVPR, 2019.
[8] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. Learning so (3) equivariant representations with spherical cnns. In ECCV, 2018.
[9] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In CVPR, 2018.
[10] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In ICCV, 2019.
[11] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. In CVPR, 2020.
[12] Qiangui Huang, Weiyue Wang, and Ulrich Neumann. Recurrent slice networks for 3d segmentation of point clouds. In CVPR, 2018.
[13] Li Jiang, Hengshuang Zhao, Shu Liu, Xiaoyong Shen, ChiWing Fu, and Jiaya Jia. Hierarchical point-edge interaction network for point cloud semantic segmentation. In ICCV, 2019.
[14] Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In CVPR, 2018. 2
[15] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In CVPR, 2018.
[16] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In CVPR, 2019.
[17] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In ICML, 2019.
[18] Bo Li, Tianlei Zhang, and Tian Xia. Vehicle detection from 3d lidar using fully convolutional network. In RSS, 2016.
[19] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In ICCV, 2019.
[20] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on Xtransformed points. In NIPS, 2018.
[21] Xinhai Liu, Zhizhong Han, Yu-Shen Liu, and Matthias Zwicker. Point2sequence: Learning the shape representation of 3d point clouds with an attention-based sequence to sequence network. In AAAI, 2019.
[22] Jiageng Mao, Xiaogang Wang, and Hongsheng Li. Interpolated convolutional networks for 3d point cloud understanding. In ICCV, 2019.
[23] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In IROS, 2015.

[24] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. In NIPS, 2019.
[25] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In CVPR, 2017.
[26] Charles Ruizhongtai Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas Guibas. Volumetric and multi-view cnns for object classification on 3d data. In CVPR, 2016.
[27] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In NIPS, 2017.
[28] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. In NeurIPS, 2019.
[29] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In CVPR, 2017.
[30] Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian. Mining point cloud local structures by kernel correlation and graph pooling. In CVPR, 2018.
[31] Martin Simonovsky and Nikos Komodakis. Dynamic edgeconditioned filters in convolutional neural networks on graphs. In CVPR, 2017.
[32] Shuran Song, Fisher Yu, Andy Zeng, Angel XChang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In CVPR, 2017.
[33] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. Splatnet: Sparse lattice networks for point cloud processing. In CVPR, 2018.
[34] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In ICCV, 2015.
[35] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and QianYi Zhou. Tangent convolutions for dense prediction in 3d. In CVPR, 2018.
[36] Lyne P. Tchapmi, Christopher B. Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese. Segcloud: Semantic segmentation of 3d point clouds. In 3DV, 2017.
[37] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, Franc¸ois Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In ICCV, 2019.
[38] Benjamin Ummenhofer, Lukas Prantl, Nils Thuerey, and Vladlen Koltun. Lagrangian fluid simulation with continuous convolutions. In ICLR, 2020.
[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In NIPS, 2017.
[40] Chu Wang, Babak Samari, and Kaleem Siddiqi. Local spectral graph convolution for point set feature learning. In ECCV, 2018.
[41] Lei Wang, YuchunHuang, Yaolin Hou, ShenmanZhang, and Jie Shan. Graph attention convolution for point cloud semantic segmentation. In CVPR, 2019.
[42] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In CVPR, 2018.
[43] Weiyue Wang, Ronald Yu, Qiangui Huang, and Ulrich Neumann. Sgpn: Similarity group proposal network for 3d point cloud instance segmentation. In CVPR, 2018.
[44] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. TOG, 2019.
[45] Felix Wu, Angela Fan, Alexei Baevski, Yann N Dauphin, and Michael Auli. Pay less attention with lightweight and dynamic convolutions. In ICLR, 2019.
[46] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In CVPR, 2019.
[47] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In CVPR, 2015.
[48] Saining Xie, Sainan Liu, Zeyu Chen, and Zhuowen Tu. Attentional shapecontextnet for point cloud recognition. In CVPR, 2018.

[49] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In ECCV, 2018.

[50] Jiancheng Yang, Qiang Zhang, Bingbing Ni, Linguo Li, Jinxian Liu, Mengdie Zhou, and Qi Tian. Modeling point clouds with self-attention and gumbel subset sampling. In CVPR, 2019.

[51] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. XLNet: Generalized autoregressive pretraining for language understanding. In NeurIPS, 2019.

[52] Wira D. Mulia, Naresh Sehgal, Sohum Sohoni, John M. Acken, C. Lucas Stanberry & David J. Fritz (2013) Cloud Workload Characterization, IETE Technical Review, 30:5, 382-397, DOI: 10.4103/0256-4602.123121

[53] Zhiyuan Zhang, Binh-Son Hua, and Sai-Kit Yeung. Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics. In ICCV, 2019.

[54] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In CVPR, 2020.

[55] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. PointWeb: Enhancing local neighborhood features for point cloud processing. In CVPR, 2019.

[56] Zhao, Hengshuang, et al. "Point transformer." Proceedings of the IEEE/CVF international conference on computer vision. 2021.

[57] Axelsson, Maria, et al. "Semantic labeling of lidar point clouds for UAV applications." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021.

[58] Liu, Yonghuai, et al., eds. 3D imaging, analysis and applications. Springer International Publishing, 2020.

[59] Zyrianov, Vlas, Xiyue Zhu, and Shenlong Wang. "Learning to Generate Realistic LiDAR Point Clouds." Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIII. Cham: Springer Nature Switzerland, 2022.