Graph Convolutional Network using Self-Attention Module in Point Cloud Classification

Bing Hui Lai, King Hann Lim, Andy Zhi Sheng Yii
Department of Electrical and Computer Engineering
Curtin University Malaysia
CDT 250 98009 Miri Sarawak, Malaysia.
700037967@student.curtin.edu.my

***Abstract*—Three-dimensional (3D) point cloud classification is one of the basic processing functions in deep learning to classify point cloud into a set of user-defined classes. This paper implements the latest state-of-the-art graph-based point cloud shape classification method-PointView-GCN and proposed a novel approach by integrating self-attention into graph convolution to enhance the classification accuracy of PointView-GCN. Self-attention is inspired by the Transformer architecture which assigns an associate weight for each updated feature after every local graph convolution. GCNNs benefits from this integration when the graph updates the features' edges by selecting the most descriptive features among its k-nearest neighbourhood. The experimental results demonstrate that all variants of the integrated self-attention models consistently outperform the default PointView-GCN. Among the tested models, the best-performing architecture achieves an impressive accuracy of 93.08%, surpassing the default PointView-GCN by an improvement of approximately 1.49%. This notable enhancement in classification accuracy highlights the importance of integrating self-attention into PointView-GCN for point cloud analysis tasks.**

***Index Terms*—Deep Learning Neural Networks, Point Cloud, 3D Shape Classification, Attention Mechanism**

I. Introduction

Point cloud are sets of data containing digital three-dimensional (3D) representations of a physical object or space. Point cloud combine XYZ coordinates with RGB values [1, 2] providing rich geometrical data, shape, and scale information in recognizing 3D objects. The applications of 3D point cloud can be found in areas such as autonomous driving, robotics, virtual reality, and medical treatment. Processing and analyzing point cloud present significant challenges due to their massive size, irregularity, and sparsity [3]. Graph convolution neural network (GCNNs) is the recent state-of-the-art point cloud classification method that operates on graph-structured data where the nodes represent the data points and the edges represent the relationships between them. GCNNs utilize the graph structure to learn feature representations that capture the local and global geometric patterns of the point cloud. GCNNs have shown superior performance in point cloud classification tasks compared to other state-of-the-art methods such as PointNet [4] and PointNet++ [5]. GCNNs effectively handle the sparsity and irregularity of point cloud which makes them a suitable choice for point cloud classification. However, GCNNs primarily prioritize local neighbourhood information for feature propagation within graphs, accurately classifying point cloud requires considering both important global context and long-range dependencies. Relying on local neighbourhood information might not adequately capture the overall structure and semantic relationships present in the point cloud.

Attention mechanisms are gaining more attention, particularly in natural language processing and computer vision tasks nowadays. Attention mechanisms enable the model to enhance focusing on the most relevant parts of the input data while diminishing the irrelevant ones [6] which possess the potential to improve the global context and long-range dependencies feature information exchanged in graph convolution. The attention mechanism learns to assign a weight to each element in the input sequence, indicating its relative importance to the task at hand. The weighted sum of the input elements is then used to compute the output of the attention layer. This can significantly improve the model's performance and efficiency by reducing the amount of computation needed to process the input data. Self-attention is a variant of attention mechanisms that have been introduced recently in transformer-based models. Self-attention allows the model to attend to different parts of the input sequence and capture the contextual relationships between them [7]. Self-attention computes the attention weights based on the input sequence itself, rather than an external context vector. This enables the model to learn the most informative representations of the input sequence while capturing the relationships between the different elements in the sequence. Self-attention has shown remarkable performance in natural language processing tasks such as machine translation and sentiment analysis [8]. In point cloud classification, attention mechanisms can be particularly useful due to the irregular and sparse nature of the data. Point cloud contain millions of points, and sometimes only a few of them may be relevant for classification. Attention mechanisms enable the model to attend to the most informative points while ignoring the irrelevant ones. This can significantly improve the model's performance and reduce the computation needed to process the point cloud.

In this paper, attention mechanisms particularly self-attention is integrated with the latest state-of-the-art graph convolution method-PointView-GCN to improve the point cloud shape classification by enhancing the feature information extraction specifying local neighbourhood and long-range dependencies. Attention mechanisms enable the model to learn more informative node representations by attending to the most relevant neighbours while disregarding the irrelevant ones in GCNNs. By integrating attention mechanisms into GCNNs, the model can learn more informative graph features node representations that capture the relevant context of the graph. Overall, this paper contributes to the research on point cloud classification by proposing an attention-based graph convolutional neural networks that achieves state-of-the-art performance. The results of this paper provide insights into the effectiveness of attention mechanisms in graph convolutional neural networks for point cloud classification which can guide future research in this area.

II. Attention module Impacts on PointView-GCN and Architectures

PointView-GCN as a graph convolution model is founded to have limitations on its reliance on local neighbourhood information, which might not effectively capture the broader context and long-distance relationships essential for precise classification. Non-local message passing in PointView-GCN only operates at a global level to update the overall feature information and its direct influence on individual local graph convolutions is limited. By incorporating attention mechanisms into graph convolution, the model gains the ability to selectively emphasize relevant parts of the input data while attenuating the irrelevant ones. Attention mechanisms have demonstrated their effectiveness in capturing global context and long-range dependencies in various domains, such as natural language processing and image analysis. By leveraging attention, the model enhances its understanding of the intricate relationships and semantic connections within the point cloud. The integration of attention mechanisms in graph convolution enables the model to allocate computational resources effectively, focusing on the most important regions of the data. This facilitates the exchange of feature information across the graph, allowing the model to capture the spatial relationships and structural dependencies more effectively. Consequently, attention-based graph convolution holds great potential to overcome the limitations of PointView-GCN and improve the accuracy of point cloud classification by leveraging global context and long-range dependencies.

A. Scaled Dot-Product Attention

The self-attention mechanism integrated into PointView-GCN is known as "Scaled Dot-Product Attention" [6] as introduced in the original transformer model by Vaswani et al. The input consists of queries and keys of dimension d_k, and values of dimension d_v. The weights of values are obtained by computing the dot products of the query with all keys, dividing each by $\sqrt{d_k}$. The dot products are scaled by $1/\sqrt{d_k}$ to counter the effect where the dot products magnitude is getting larger for larger values of d_k, pushing the softmax function into the regions where it has extremely small gradients. The queries, keys, and values are packed together into matrix Q, K, and V to compute the attention function as:

$$\text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}\left(\frac{\mathbf{QK}^T}{\sqrt{d_k}}\right)\mathbf{V}, \qquad (1)$$

B. Multi-Head Attention

Instead of using a single attention function with d_{model}-dimensional keys, values, and queries, the self-attention layer is performed in multi-head attention architecture [6] where the queries, keys, and values are linearly projected h times with distinct learning linear projection to d_k, d_k, and d_v dimensions respectively. The projected versions of queries, keys, and values are fed into a parallel attention function, resulting a d_v-dimensional output values. These outputs are concatenated and projected again to obtain the final values. The model is able to jointly attend to information from different representation sub-spaces at different positions as follows,

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1,\ldots,\text{head}_h)\mathbf{W}^O, \qquad (2)$$

$$\text{head}_i = \text{Attention}(Q\mathbf{W_i^Q}, K\mathbf{W_i^K}, V\mathbf{W_i^V}), \qquad (3)$$

where the projections are parameter matrices $\mathbf{W_i^Q} \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $\mathbf{W_i^K} \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $\mathbf{W_i^V} \in \mathbb{R}^{d_{\text{model}} \times d_v}$, and $\mathbf{W^O} \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.

III. Self Attention in PointView-GCN

At first, the shape features are extracted with backbones using PointNet++ [5] for every captured single view point cloud data (PCD). An initial view graph $G = \{v_i\}_{i\in N}$ of N nodes is constructed where each node v_i is represented by the shape features F_i corresponding to each single-view PCD. Firstly, local graph convolution updates the features of each node v_i^j at each graph level G_j considering its neighbouring nodes as

$$F^j = L(\mathbf{A}^j\mathbf{F}^j\mathbf{W}^j;\alpha^j), \qquad (4)$$

A multi-head self-attention is added after each local graph convolution in PointView-GCN. The updated features F^j are passed into a multi-head self-attention layer which takes the query, key, and value as input and represented by the updated features F^j as:

$$F^j_{attn} = \text{MultiHead}(F^j, F^j, F^j), \qquad (5)$$

The resulting output is denoted as F^j_{attn}, contains the associated weights for every feature at each node v_i^j at the graph level G^j. This allows a more descriptive representation of the point cloud data, facilitating improved feature learning and classification. The multi-head self-attention employed a number of heads, h = 4 where four attention layers operate in parallel. Each head employs a reduced dimension of $d_k = d_v = d_{model}/h = 128$ as the extracted features dimension of F^j = $d_k = d_v = 512$. As a result of the reduced dimensionality, the overall computational expense is comparable to that of single-head attention with full dimensionality. The attention output features $F^j_{multihead}$ is added into F^j to capture the interdependencies between the features and nodes, resulting in

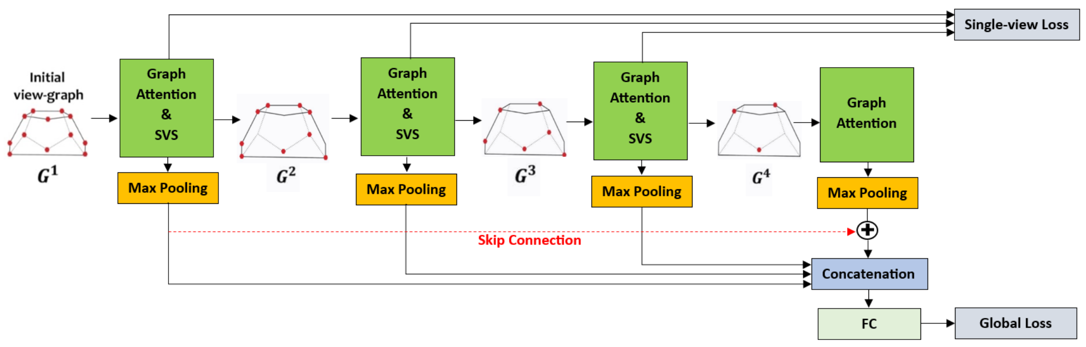

Fig. 1. Implementation of multi-head self attention in PointView-GCN [9].

a more detailed and contextually-aware feature representation as follows,

$$F^j_{attn} = F^j_{multihead} + F^j. \tag{6}$$

The resulting output F^j_{attn} is fed into a layer normalization, inspired by the transformer [10] architecture to ensure the distribution of the activation between layers remains consistent throughout the neural network, improving the stability and performance of the model. The layer normalization of the attention output features (LayerNorm(F^j_{attn})) combined with the initial features can be expressed as,

$$F^j_{attn} = \gamma \frac{F^j_{attn} - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta, \tag{7}$$

where μ and σ are the mean and variance of the sample F^j_{attn} across all its hidden units respectively, γ and β are learnable parameters for scaling and shifting the normalized output. ϵ is a small positive constant added for numerical stability. Features $\mathbf{F}^j_{attn}$ are further updated by considering the long-range relationships among all nodes using Non-local message passing. Each node v_i first updates the state of its edge with neighbouring node v_j as a message

$$m^j_{i,p} = R(F^j_{i,attn}, F^j_{p,attn}; \beta^j)_{i,p \in N^j} \tag{8}$$

where $R(.)$ is the relation functions among pairs of views with the related parameters β^j. The feature of node v^i is then updated by itself and all the received pair-wise messages as

$$F^j_{i,attn} = C(F^j_{i,attn}, \sum_{p=1,p\neq 1}^{N_j} m^j_{i,p}; y^j) \tag{9}$$

Selective view sampling (SVS) is then applied to select the most descriptive view among each neighbourhood. The view graph G^j is first sampled using Farthest Point Sampling (FPS). Among the k-nearest neighbourhood (KNN), $\mathbf{V}^j_i$, of each sample node v_i, a view-selector is applied to select the node with the highest response of the softmax function. Finally, a new coarsened view-graph G^{j+1} with its associated updated feature F^{j+1}_{attn} is being fed to the next level of graph convolution. Fig. 1 shows the Graph Attention Layer with an integration of the proposed multi-headed self-attention into the PointView-GCN architecture. After every local graph convolution with multi-headed self-attention, a max-pooling is performed on the updated features $\mathbf{F}^j_{attn}$ to obtain a global shape feature $F^j_{global,attn}$ at each level. One residual connection is added from the output of the first convolution level to the last one to prevent vanishing gradients when the number of GCN levels increased as shown in Fig.1. The training loss consists of two parts in this method which are global shape loss L_{global} and selective-view shape loss $L_{selective}$:

$$L = L_{global}(S(F_{global}, y) + \sum_{j=1}^{M} \sum_{i=1}^{N^{j+1}} \sum_{v_s \in V^j_i} L_{selective}(V(F^j_s; \theta^j), y) \tag{10}$$

where L_{global} is the cross-entropy loss, S is a classifier with a fully connected layer and softmax function, y is the shape category. $L_{selective}$ is the cross-entropy loss for the view selector. $V(.)$ is the function for the view selector with parameter θ^j, and F^j_s is the feature of the sub-sampled node.

IV. Experimental Setup

The experiments are conducted on a computer with an AMD Ryzen 5 3550H with Radeon Vega Mobile Gfx 2.10 GHz CPU. The system is equipped with 12 GB of DDR4 RAM operating at a clock speed of 2400 MHz, and a NVIDIA Geforce GTX1650 graphics card. The benchmark synthetic point cloud dataset - ModelNet40 [11] is used in the experiment to evaluate the classification accuracy of the self-attention PointView-GCN model. ModelNet40 consists of 12311 mesh models from 40 different object categories. 9843 models about 80% is randomly sampled for training and 2468 (20%) models are used for testing. A random seed is set to 9900 for the random number generators to ensure the output result is reproducible. Two experiments are conducted in this paper to evaluate the effectiveness of the proposed method. The first experiment is to evaluate the effectiveness of integrating self-attention in four specific layers of local graph convolution. The second experiment is conducted to test the overall performance of the model by comparing the performance of the model with and without self-attention.

There are a few evaluation metrics used in the experiment to evaluate the performance of the Enhancing PointView-GCN with self-attention model. Training accuracy is the most commonly used evaluation metric in classification tasks. It measures the portion of the correctly predicted samples made by the model for the same training samples. Accuracy is calculated as:

$$Acc = \frac{S_c}{S_p}, \tag{11}$$

where Acc is defined as the accuracy, S_c is the amount of correctly predicted samples, and S_p is the amount of samples. Loss is another commonly used metric in classification tasks. Loss measures how well the model can minimize the difference between the predicted output and the actual output. In the experiments, Cross-entropy loss is utilized as it is an effective approach for training models in classification tasks. The equation for Cross-entropy loss can be written as:

$$\begin{aligned} \text{Loss}(x, class) &= -\log\left(\frac{\exp(x[class])}{\sum_j \exp(x[j])}\right) \\ &= -x[class] + \log\left(\sum_j \exp(x[j])\right), \end{aligned} \tag{12}$$

where x is the input to the model, and class is the index of the target class. The loss function computes the negative log-likelihood of the predicted class probability distribution, where the softmax function is applied to the input to obtain the predicted probabilities. Besides, there are two testing accuracy evaluation metrics used to assess the performance of the model which are validation mean accuracy and validation overall accuracy. Validation means accuracy measures the average accuracy of all classes in the validation dataset. The equation for validation mean accuracy is as follows:

$$mAcc = \frac{1}{n_c}\sum_{i=1}^{n_c} \text{acc}_i, \tag{13}$$

where $mAcc$ is the mean accuracy, n_c is the number of classes and acc_i is the accuracy of class i. Validation overall accuracy is the ratio of the total correctly classified samples over the total number of samples in the validation dataset. The equation for validation overall accuracy is as follows:

$$\text{OA} = \frac{T_c}{T_p}, \tag{14}$$

where OA is the overall accuracy, T_c is the total number of correct predictions, and T_p is the total number of predictions.

V. Experiment #1: Study of Attention Layer on PointView-GCN

PointView-GCN [9] consists of four layers of local graph convolution. This experiment investigates the impact of integrating the self-attention mechanism in PointView-GCN architecture. There are four types of architectures proposed in this experiment, each with a specific number of self-attention layers implemented in PoinView-GCN. The first architecture has self-attention only in the first layer called GA-1. The second architecture is GA-2 which has self-attention in the first two layers. The third architecture, GA-3 has self-attention in the first three layers. The fourth and final architecture, GA-4 has self-attention in all four layers. These proposed architectures are tested by the ModelNet40 [11] full dataset for 20 epochs. The training loss, training accuracy, and testing accuracy are recorded for comparison and to determine the optimal number of self-attention layers for PointView-GCN.

Fig. 2 and Fig. 3 shows the validation mean and overall accuracy of the four integrated self-attention PointView-GCN architectures which classify the ModelNet40 dataset excluding the training dataset with 20 epochs. It is observed that GA-3 outperforms the other architecture showing the highest validation mean and overall accuracy around 91.07% and 93.08% respectively. GA-3 demonstrated the highest validation accuracy indicating its superior ability to generalize to unseen and new data. Hence, GA-3 is selected as the best model in this experiment.

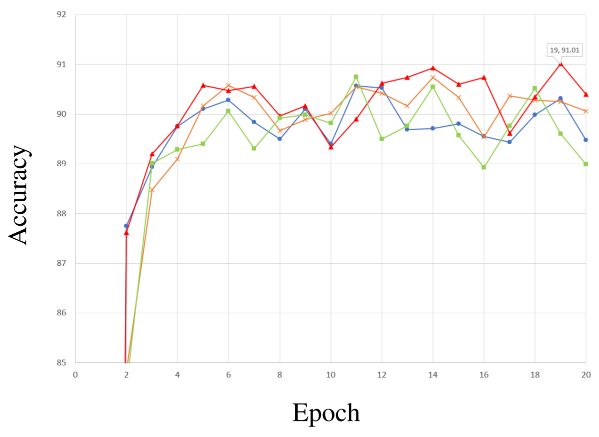

Fig. 2. Validation Mean Accuracy of Graph Attention models. Blue line circle marker (GA-1), Orange line cross marker (GA-2), Red line triangle marker (GA-3), and Green line rectangular marker (GA-4).

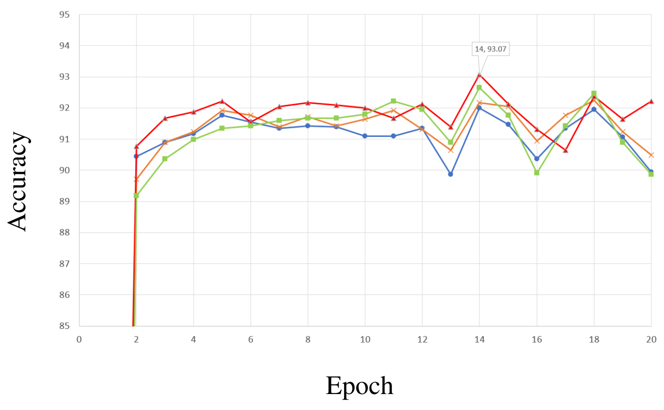

Fig. 3. Validation Overall Accuracy of Graph Attention models. Blue line circle marker (GA-1), Orange line cross marker (GA-2), Red line triangle marker(GA-3), and Green line rectangular marker (GA-4).

VI. Experiment #2: Performance Evaluation of Proposed Graph Attention PointView-GCN

The performance of the proposed integrated self-attention PointView-GCN model is evaluated using the ModelNet40 benchmark dataset and compared to the default PointView-GCN model. Table I shows the performance comparison of these models in terms of training and testing metrics. In Table I, it is evident that the proposed graph-attention models outperform the default PointView-GCN model. Among these models, the top-performing GA-3 reduces training losses by 0.26 compared to PointView-GCN. GA-3 also achieves a higher testing mean accuracy by 0.4% and testing overall accuracy by 1.08% from the PointView-GCN model. It should be noted that both graph-attention models exhibit slightly lower training accuracy than PointView-GCN, except for GA-2 which improves training accuracy by 0.3%. It performs the best and improves from PointView-GCN when testing with the training dataset.

The training accuracy is a measure of how well the model fits the training data, while the testing accuracy is a measure of how well the model generalizes to new, unseen data. The fact that both GA-1, GA-3, and GA-4 models have slightly lower training accuracy than PointView-GCN suggests that these models may be overfitting to the training data. However, their higher testing accuracy indicates that they are better able to generalize to new data, which is the main goal in the paper for the point cloud classification model. Overall, GA-3 offers superior performance compared to other graph attention models and successfully improved the classification accuracy of PointView-GCN. It is explainable why GA-3 offers the best performance as the integrated self-attention benefits PointView-GCN from SVS and Non-local message passing which updates the nodes' features with its neighbouring nodes and selects the most descriptive node features forming a new graph in the next level. The last layer of GCN does not involve SVS and Non-local message passing thus GA-3 provides the best performance compared to GA-4. GA-3 also benefits from a reduction in model complexity by adding self-attention only in the first three layers, preventing the overfitting effect.

VII. Conclusion

PointView-GCN as a graph convolution model is founded to have a limitation on its reliance on local neighbourhood information. This paper explores the integration of self-attention in the four specific layers of local graph convolution in the PointView-GCN model. This approach enables the model to selectively attend to relevant information from neighbouring nodes and dynamically weigh their contributions during the feature aggregation process. The integrated self-attention models demonstrate a notable improvement in classification accuracy compared to the original PointView-GCN model. The best-performing model, GA-3, which incorporates self-attention in the first three layers, achieves the highest testing classification accuracy of 93.08%, surpassing the default PointView-GCN model by 1.49%. The validity of this improvement is verified through various evaluation metrics, including losses, confusion matrix, precision, recall, and F1 score. The successful implementation of self-attention demonstrates the effectiveness of incorporating attention mechanisms in graph convolutional networks to improve classification accuracy.

As a promising avenue for future research and development, the exploration of attention mechanisms holds great potential for enhancing and extending the current project. Attention mechanisms have demonstrated their effectiveness in point cloud classification using graph convolution. An intriguing research direction involves exploring the potential of three SOTA attention mechanisms: Co-attention, Hierarchical attention, and Multi-dimension attention [12]. Each type offers distinct advantages and characteristics that can contribute to improving the model's performance and capabilities.

TABLE I
Performance comparison on integrated self-attention GCN.

Methods	Training Loss	Training OA (%)	Testing mAcc (%)	Testing OA (%)
PointView-GCN	1.4064	98.71	89.17	91.59
GA-1	1.3593	98.67	90.57	92.00
GA-2	1.3505	99.02	90.74	92.25
GA-3	1.1443	98.57	91.01	93.08
GA-4	1.3481	98.64	89.60	92.66

References

[1] Y. Guo, F. Sohel, M. Bennamoun, M. Lu, and J. Wan, "Rotational projection statistics for 3d local surface description and object recognition," *International journal of computer vision*, vol. 105, no. 1, pp. 63–86, 2013.

[2] H. Lu and H. Shi, "Deep learning for 3d point cloud understanding: a survey," *arXiv preprint arXiv:2009.08920*, 2020.

[3] A. Komarichev, Z. Zhong, and J. Hua, "A-cnn: Annularly convolutional neural networks on point clouds," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 7421–7430.

[4] 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*, 2017, pp. 652–660.

[5] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[7] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[8] P. Kumar, K. Pathania, and B. Raman, "Zero-shot learning based cross-lingual sentiment analysis for sanskrit text with insufficient labeled data," *Applied Intelligence*, pp. 1–18, 2022.

[9] S. S. Mohammadi, Y. Wang, and A. Del Bue, "Pointview-gcn: 3d shape classification with multi-view point clouds," in *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2021, pp. 3103–3107.

[10] R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, and T. Liu, "On layer normalization in the transformer architecture," in *International Conference on Machine Learning*. PMLR, 2020, pp. 10 524–10 533.

[11] J. Sun, Q. Zhang, B. Kailkhura, Z. Yu, and Z. M. Mao, "Modelnet40-c: Arobustness benchmark for 3d point cloud recognition under corruption."

[12] Z. Niu, G. Zhong, and H. Yu, "A review on the attention mechanism of deep learning," *Neurocomputing*, vol. 452, pp. 48–62, 2021.