2024 IEEE International Conference on Multimedia and Expo (ICME) | 979-8-3503-9015-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICME57554.2024.10687460

PointELM: Fast Point Cloud Classification Using Deep Random Mapping Based Extreme Learning Machines

1st Zhuangzi Li
School of Electronic and Computer Engineering
Peking University
Shenzhen, China
lizhuangzii@163.com

2nd Shan Liu
Media Lab
Tencent
Palo Alto, USA
shanl@tencent.com

3rd Ge Li ✉
School of Electronic and Computer Engineering
Peking University
Shenzhen, China
geli@ece.pku.edu.cn

***Abstract*—Designing an influential and instructive deep network has become a prominent research in point cloud analysis. However, deep networks inevitably require extensive training time and are sensitive to variational data distribution. In this paper, we observe that a randomly initialized network exhibits discriminative abilities and show training deep networks is not necessary for point cloud classification. Specifically, we propose a Point Extreme Learning Machine (PointELM), which initially extracts infant features from point clouds using a randomly weighted network. Subsequently, a frequency-domain mapping is designed to enhance the infant features. Finally, an ELM classifier is adopted to categorize the enhanced features and generate the output predictions. We evaluate PointELM on classical networks and highlight three advantages: (1) PointELM does not require backpropagation, enabling extremely fast training. Yet PointELM can still achieve promising performance. For example, the classification accuracy of a DGCNN-based PointELM just lowers a trained DGCNN about 2.8% on ModelNet40. (2) PointELM is a flexible method that can conveniently transfer a trained network to a new dataset with minimal performance loss. (3) PointELM can rapidly alleviate the performance degradation caused by feeding sampled point clouds to trained networks, indicating strong potential on adjusting data with various distributions. Code is available at *https://github.com/lizhuangzi/PointELM*.**

***Index Terms*—Point Cloud, Extreme Learning Machine, Deep Learning**

I. Introduction

Recently, the development of the multimedia industry has experienced the flourishing of 3D point cloud in many fields like virtual reality, satellite, and automatic pilot [1]. Point cloud classification is a fundamental task in the multimedia community, which yields many influential architectures for deep networks. The PointNet series [2], [3] set a precedent for point-wise architectures and guarantee the permutation invariance of point clouds. Then, many researchers [4]–[6] explore ways to better capture the local receptive field in convolutional neural networks to convolve local features. Particularly, Wang et al. [5] present the dynamic graph convolution for building topology on point clouds, which establishes a popular network architecture for many point cloud tasks. Li et al. [7] try to increase the depth of networks, but the over-smoothing problem in graph neural networks hinders the performance to a great extent. Hence, some follow-up works attempt to design powerful structures, e.g., Transformer architectures [6], MLP modules [8] and attention-based mechanisms [9]. On the other hand, Qian et al. [10] explore the optimizing strategy of networks and propose PointNeXt, showing the importance of rational experimental settings.

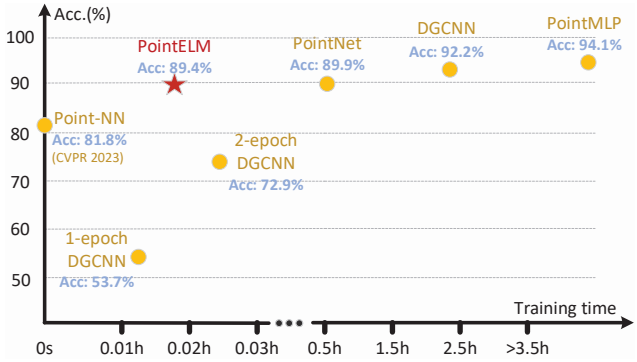

Fig. 1: Training time and testing accuracy on ModelNet40. The PointELM adopts an untrained DGCNN [5] as the backbone. The "1-epoch DGCNN" denotes training a DGCNN using one-epoch data. Some classic methods, i.e., PointNet [2], PointMLP [8] and Point-NN [11] are introduced.

Although the above-mentioned methods present delicate modules and achieve satisfying performance, they take a lot of effort to adjust networks' parameters and usually waste time on waiting training. Besides, according to the investigation of [12], many point cloud classification networks suffer from performance degradation when point clouds are sampled. To solve the problem, Zhang et al. [11] propose a parameterless point cloud processing framework named Point-NN without training phases, enabling quick and easy classification of point clouds. However, the classification performance of Point-NN is restricted, and its inference speed is slower than a

✉ Corresponding author: Ge Li. This work is supported by National Natural Science Foundation of China (No. 62172021).

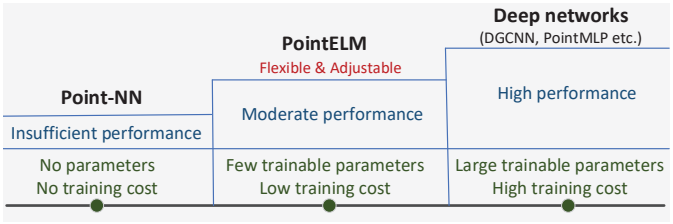

Fig. 2: The proposed PointELM bridges the gap between Point-NN and other deep networks.

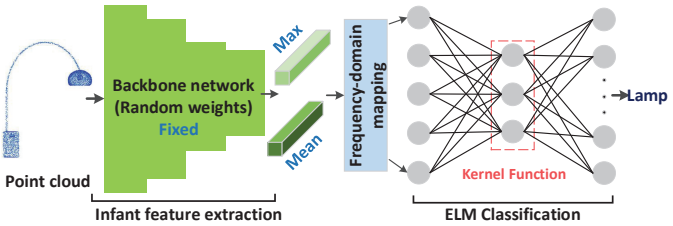

Fig. 3: The overall inference pipeline of PointELM.

common neural network. On the other hand, we observe Point-NN employs farthest point sampling, K-nearest neighbor, and pooling algorithms, which are usually available in a point cloud analysis network. Hence, it enlightens us to explore whether a randomly weighted deep network has the capability to extract valuable features. Coincidentally, in machine learning, Extreme Learning Machine (ELM [13]) shows a single-hidden layer neural network with random weights can be used for classification. Therefore, we introduce ELM to exploit the potential of deep random mapping and propose a novel PointELM framework.

PointELM first extracts infant features from point clouds via a backbone network with random weights. Subsequently, a frequency-domain mapping is formulated to convert the infant features to the frequency domain. Lastly, we employ the kernel ELM classifier to effectively classify the frequency-domain features and generate predictive outcomes. Due to the backbone network not needing to be trained, PointELM can be performed extremely fast. As shown in Fig. 1, we take about 2.5 hours to train a DGCNN with 92.2% accuracy, but a randomly weighted DGCNN as the backbone of PointELM can achieve 89.4% accuracy with only 0.02 hours of training time. Furthermore, as shown in Fig. 2, PointELM bridges the gap between non-parametric and deep methods with few trainable parameters and low training cost. Experiments also demonstrate PointELM's flexibility and adjustability on different networks and data distributions. Moreover, our studies also reveal that a randomly weighted network itself has strong feature extraction capability, and the capability highly depends on how to construct it.

II. Our Approach

Our inference pipeline, illustrated in Fig. 3, comprises three stages: infant feature extraction, frequency-domain mapping, and ELM classification. During the training phase, the feature extraction and frequency-domain mapping can provide a feature matrix from training data, and only the ELM classifier is trained. Hence, PointELM operates independently of backpropagation, allowing the extraction of infant features via CPU. Detailed algorithms are illustrated in following sections.

A. Infant feature extraction with random weights

It is widely believed that an untrained network lacks the ability to extract meaningful feature representations. However, our observation reveals that globally extracted features from a randomly initialized DGCNN [5] exhibit discernible separations when visualized using TSNE [14]. As shown in Fig. 4 (a), the TSNE visualization shows that the three feature categories are not entirely intertwined; rather, they exhibit distinct regions in the feature space. This observation suggests that untrained point cloud networks retain a degree of feature extraction capability. Therefore, a randomly weighted DGCNN (notated as $F(\cdot)$) is exploited as the backbone for PointELM. Let's note N point clouds $\mathbf{P} = \{\mathbf{p}_1, ..., \mathbf{p}_N\}$, and each $\mathbf{p} \in \mathbb{R}^{n\times 3}$ represents a 3D point cloud with n points. Features after global max-pooling and average pooling are concatenated to 2048-dimensional infant features $\mathbf{X} \in \mathbb{R}^{N\times 2048}$ as:

$$\mathbf{X} = [\mathrm{Max}(F(\mathbf{P})), \mathrm{Mean}(F(\mathbf{P}))]. \tag{1}$$

Similarly, other networks such as PointNet and PCT are viable options for PointELM, allowing us to employ an analogous strategy to get $N \times 2048$ concatenated features. Note that the term "random weights" extends beyond random initialization for the backbone network; it can also encompass training the backbone network on comparable or unrelated datasets. As a result, our work is different with [15], because the backbone of [15] needs to be trained.

B. Frequency-domain mapping

Tancik et al. [16] manifest converting feature to trigonometric representation benefits high-frequency learning. Therefore, we aim to excavate frequency-domain information of the infant features, so as to promote their discriminability. Motivated by it, we propose the frequency-domain mapping $M : \mathbb{R}^{n\times 2048} \rightarrow \mathbb{C}^{n\times d}$, where features are mapped through trigonometric functions, formulated by:

$$\mathbf{X}^{'} = M(\mathbf{X}) = \mathrm{FFT}(\mathbf{X})e^{-i2\pi\mathbf{B}}, \tag{2}$$

where $F(.)$ denotes the Fourier transform, and $\mathbf{B} \in \mathbb{R}^{2048\times d}$ is a random angle matrix initialized by Gaussian. The $e^{-i2\pi\mathbf{B}}$ can be implemented by Euler's formula via complex multiplication as illustrated in [17]. For convenience, we consider $\mathbf{X}^{'}$ as two concatenated vectors, represented by $\mathbf{Z} \in \mathbb{R}^{N\times m}$, where $m = 2d$. Frequency-domain mapping enhances the diversity of feature representations but inevitably introduces additional computational overhead. As shown in Fig. 4 (b), the frequency-domain mapping expands the distances between features, potentially enhancing the classifier's learning.

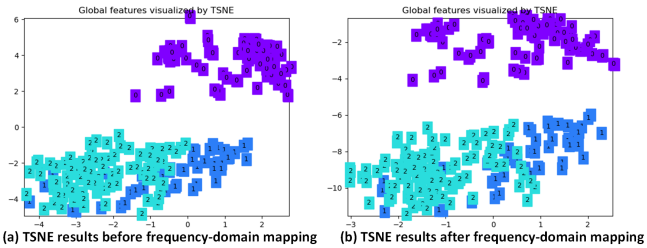

Fig. 4: TSNE [14] visualization results of infant features.

C. ELM classification

The original of ELM is a particular single-layer feed-forward network [13], known for its rapid fitting of data and label pairs using the least square method. In this work, we feed the mapped frequency-domain features $\mathbf{Z} \in \mathbb{R}^{N\times m}$ into an ELM classifier. We denote a random weight matrix $\mathbf{W} \in \mathbb{R}^{m\times L}$, where L is the number of hidden nodes, the ELM mapping $\mathrm{H}(\mathbf{Z})$ is formulated as:

$$\mathbf{H} = \mathrm{H}(\mathbf{Z}) = \mathrm{H}(\mathbf{Z}\mathbf{W} + \mathbf{b}^T), \tag{3}$$

where $\mathbf{b} \in \mathbb{R}^{L\times 1}$ is the bias vector, and $\mathrm{H}(.)$ is the activation function. As we obtain $\mathbf{H}$, the optimization objective of ELM for classification can be written by solving the following minimization problem:

$$\arg\min_{\alpha} \quad \|\mathbf{Y} - \mathbf{H}\alpha^T\|_2^2, \tag{4}$$

where $\mathbf{Y} \in \mathbb{R}^{N\times c}$ is the label matrix of the training data, and c denotes the number of category. The $\alpha \in \mathbb{R}^{c\times L}$ represents a trainable weight matrix that linearly combines the outcomes of the ELM mapping. To compute α, the standard least-squares method is utilized to obtain the analytical solution:

$$\hat{\alpha}^T = \mathbf{H}^T(\mathbf{H}\mathbf{H}^T)^{-1}\mathbf{Y}, \tag{5}$$

As $\mathbf{H}\mathbf{H}^T$ is typically nonsingular, the term $\mathbf{H}^T(\mathbf{H}\mathbf{H}^T)^{-1}$ can be approximated by a Moore-Penrose generalized inverse matrix $\mathbf{H}^\dagger$ to expedite training. During the testing phase, $\hat{\alpha}^T$ is directly employed for inference and prediction:

$$\hat{\mathbf{Y}} = \mathrm{H}(\mathbf{Z}_{test})\hat{\alpha}^T, \tag{6}$$

where $\mathbf{Z}_{test}$ is the matrix of infant features from testing data, and $\hat{\mathbf{Y}}$ denotes the predicted classification result.

This study considers utilizing a kernel ELM, a specialized form of ELM mapping, which replaces $\mathrm{H}(\mathbf{Z})$ with a Radial Basis Function (RBF). The RBF is more potent as it can map features into an infinite-dimensional space, formulated as follows:

$$\mathrm{H}(\mathbf{C}, r, \gamma, \mathbf{Z}) = \exp\left(\frac{-\gamma d(\mathbf{Z}, \mathbf{C})}{r^2}\right), \tag{7}$$

where the $\mathbf{C} \in \mathbb{R}^{m\times L}$ is centers of RBF, which are uniformly initialized over the input space or use examples from data as the center. $d(.)$ is the function that computes distance between each pair of the two inputs. r is the radii of the $\mathbf{C}$. The γ, set to 10^{-3}, is a hyperparameter used to regulate intensity.

TABLE I: Training and testing MSEs ($\times 10^{-2}$) with different backbones for PointELM.

MSE	PointNet	PointNet++	PCT	PointMLP	DGCNN
Training	1.79	3.37	1.82	5.20	1.77
Testing	3.84	4.70	3.75	9.22	3.50

D. Generalization analysis

Generalization reveals the performance of a trained model on unseen datasets. Here, we directly give the generalization bound of PointELM measured by Rademacher complexity [18].

Theorem 1: Let $\mathcal{H}$ be a family of functions of PointELM, for all $h \in \mathcal{H}$ we define generalization error $\mathcal{L} = \{x \mapsto |h(x) - y|^2\}$ which is the family of bounded loss functions $[0, K]$ associated to ground-truth label y. Besides, $\mathcal{R}_S(\mathcal{H})$ is the empirical Rademacher complexity of $\mathcal{H}$ over N training samples. Then, for any $\sigma > 0$, with probability at least $1 - \sigma$. The following inequality holds for all $h \in \mathcal{H}$:

$$\mathrm{E}[\mathcal{L}] \leq \frac{1}{N}\sum_{i=1}^{N}|h(x_i) - y_i|^2 + 2K\mathcal{R}_S(\mathcal{H}) + 3K\sqrt{\frac{\log\frac{2}{\sigma}}{2N}}. \tag{8}$$

The theorem indicates small $\mathcal{R}_S(\mathcal{H})$ and large N can guarantee generalization. As shown in Table. I, the features extracted by DGCNN have the smallest training error and the Rademacher complexity of them is closed (See Supp.), thus exhibiting the lowest generalization error on the testing set.

III. Experiments

A. Dataset and experimental setting

To demonstrate the effectiveness of our approach, we employ four classical datasets in our experiments, i.e., ModelNet40 [19], ScanObjNN [20], ShapeNet [21] and Furniture [22]. The ModelNet40 dataset contains 40 categories, following the official split with $9,843$ shapes for training and $2,468$ for testing. The ScanObjNN dataset comprises 15 categories, and we use the classical split modes described in [11] for training and testing. The ShapeNet dataset includes 50 categories, and we adhere to the official split, utilizing $13,998$ instances for training and $2,874$ for testing. The Furniture dataset is a recent 3D texture dataset offering two split modes: a super mode with 8 categories and a detailed mode containing 50 categories. It has $11,592$ instances for training and $2,484$ for testing. During the training phase, we extract features from all samples in the training set using the backbone network, without employing data augmentation. The weights of the backbone are initialized using Kaiming normal [23] by default. The experiments are performed on the Ubuntu 16.04 system, utilizing an Intel(R) Xeon(R) Gold 5115 CPU @ 2.40GHz. Experiments involving other deep networks are carried out on a Tesla V100 GPU with CUDA version 11.0.

TABLE II: Classification accuracy (%) and running time comparisons of different approaches on ModelNet40 [19].

Method	Overall Acc (%)	Class Acc (%)	Training Time (h)	Testing Time (s)
PointNet [2]	89.9	86.3	0.6	–
DGCNN [5]	92.2	89.3	2.4	–
PCT [6]	93.4	90.4	2.9	–
PointMLP [8]	94.1	–	14.4	–
ELM-LRF [24]	83.6	79.7	0.03	29.3
Point-NN [11]	81.8	78.6	0.00	65.7
PointELM w/o. F	89.4	84.7	0.01	15.6
PointELM	89.4	85.8	0.02	20.5

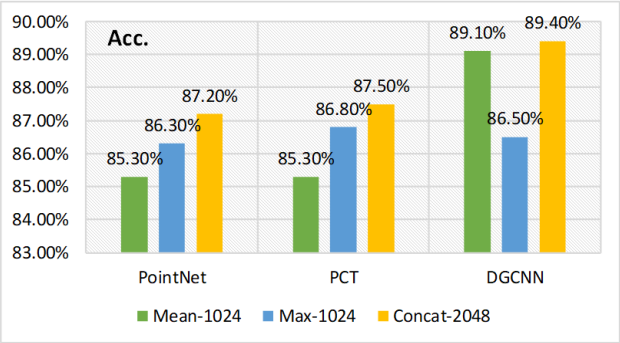

Fig. 5: Investigating infant features of different backbones.

B. Experiments on ModelNet40

First, we compare PointELM with several other point cloud classification methods on ModelNet40, including representative networks such as PointNet [2], DGCNN [5], PCT [6], and PointMLP [8]. We modify ELM-LRF [24] on point cloud task, which integrates the local receptive field component from [3] with an ELM classifier. Additionally, Point-NN [11], a non-parametric classification method requiring no training, serves as an important comparison method. PointELM adopts a randomly initialized DGCNN as its backbone. We denote "PointELM w/o. F" as the version of PointELM excluding the frequency-domain mapping. As shown in Table II, "PointELM w/o. F" achieves 89.4% accuracy and only takes 0.02 h for training. Comparing to Point-NN, PointELM gains 7.6% improvements with low training cost. The frequency-domain mapping brings more training and testing time, but it outperforms PointELM by 1.1% class accuracy. Here, the number of hidden layers of PointELM is set to 2048 ($L = 2048$) and $d = 1250$ in the frequency-domain mapping.

We then investigate the infant features of different backbones. As shown in Fig. 5, concatenating 2048-dimensional features after max pooling and mean pooling achieves the best performance, demonstrating uniting global features as Eq. (1) is effective. Besides, the findings suggest that the features extracted by DGCNN are more suitable for PointELM compared to PCT and PointNet. It also manifests that the performance of PointELM relies on how to design a network.

Next, we compare ELM classifier with other classifiers to show its advantage. In the experiment, we respectively configure the numbers of hidden layers are 1000 and 2000, using the Sigmoid activation function as defined in Eq. (3). As shown in Table. III, other classifiers like linear SVM, kernel SVM, and the random forest do not surpass ELM in performance. Notably, the 2000-layer ELM, requiring only 2.6 s for training, achieves an accuracy of 88.4%, outperforming other classifiers. As a result, ELM can be regarded as a promising discriminator for infant features.

TABLE III: Investigation of different classifiers on ModelNet40.

Indicator	Linear SVM	Kernel SVM	Random Forest	ELM L=1000	ELM L=2000
Acc(%)	67.1	70.8	84.0	86.3	88.4
Train time	56.7 s	53.6 s	64.4 s	1.0 s	2.6 s

TABLE IV: Classification results of handling sampled points with different ratios on the ModelNet40 testing set; The evaluation network is a trained DGCNN with 1024 points.

Method	$n = 512$	$n = 256$	$n = 128$	$n = 64$	$n = 32$
None	70.0	34.5	9.9	4.5	2.2
Point-NN [11]	80.6	75.6	62.8	46.6	4.5
SPU [12]	90.9	88.1	78.4	59.7	–
SampleNet [25]	82.6	45.8	11.5	7.9	–
PointELM w/o. F	90.3	89.6	85.9	83.6	77.8
PointELM	91.0	89.6	86.9	84.0	78.4

C. Experiments on sampled point clouds

Li et al. [12] indicate sampling easily changes points' distribution, leading to decreased classification performance on a well-trained network. Hence, ensuring a network's accurate classification of sampled point clouds becomes crucial. Except for retraining a new model, addressing this issue typically involves two ways: (1) Developing an upsampling method to recover the original number of points like SPU [12]. (2) Crafting a powerful sampling strategy (e.g., SampleNet [25]) to alleviate performance degradation. We conduct experiments on a well-trained DGCNN with 1024 points as shown in Table. IV, where n denotes the number of points in a point cloud. When n reduces, we can see the performance degradation phenomenon is obvious if we do nothing (denoted by "None"). Moreover, other methods exhibit poor performance, particularly in scenarios with large sampling ratios. Fortunately, PointELM provides a third way that can fast and accurately adjust new data distributions with different sampling ratios. For the sampled point clouds, we directly employ the DGCNN to extract features of sampled point clouds. Then, the ELM classifier is adopted to learn those features with corresponding labels by Eq. (5). At $n = 32$, DGCNN achieves only 2.2% accuracy, seems showing poor feature quality. However, PointELM learns from these features, reaching an accuracy of 78.4%. When sampling ratio is small, e.g., $n = 512$, PointELM still achieves the highest accuracy among others. The experiments strongly suggest that PointELM effectively

TABLE V: Comparing PointELM with re-training and fine-tuning strategies on ShapeNet [21].

Indicator	Fine-tuning DGCNN*	Re-training DGCNN*	PointELM -DGCNN	PointELM -DGCNN*
Acc(%)	95.5	99.0	98.5	98.7
Train time	2.0 h	3.6 h	74.0 s	74.0 s

TABLE VI: Classification accuracy (%) on the shuffled ScanObjectNN testing set [20]. PCT* denotes only training PCT on the "Split-1"; Point-NN† denotes testing the model with batch size = 1 for keeping its performance.

Method	Split-1	Split -2	Split-3	Trainable Param.
PointNet [2]	73.3	79.2	68.2	3.5 M
PointNet++ [3]	82.3	84.3	77.9	1.7 M
DGCNN [5]	82.8	86.2	78.1	1.8 M
Point-NN† [11]	67.5	72.1	56.7	0.0 M
PointELM	69.7	72.8	57.4	0.03 M
PCT* [6]	83.0	75.0	64.9	2.8 M
PointELM-PCT*	83.1	83.0	71.8	0.03 M

resolves performance degradation resulting from point cloud sampling.

D. Experiments on dataset transfer

Fitting a trained model to a new dataset is intuitive for transferring. However, due to the catastrophic forgetting problem, a single network cannot perform well on both two datasets. Moreover, re-training or fine-tuning a network still needs much computational load and time. While PointELM can fast transfer a model trained on one dataset to other datasets for its flexibility. In this experiment, we leverage a pre-trained DGCNN and compare the transfer performance among the re-training strategy, fine-tuning strategy, and PointELM. The The notation DGCNN* signifies a pre-trained DGCNN on ModelNet40 transferred to the ShapeNet dataset [21]. Conversely, DGCNN denotes weights randomly initialized without pre-training. The fine-tuning strategy exhibits ineffective dataset transfer capabilities, while re-training a network on ShapeNet achieves 99.0% accuracy at the expense of a lengthy 3.6 h for training. In contrast, PointELM achieves 98.7% accuracy in a mere 74 s of training. Another experiments on ScanObjNN [20] is shown in Table. VI. "Split-1", "Split-2" and "Split-3" have different data distributions. The PCT* trained on "Split-1" exhibits poor performance on the "Split-2" and "Split-3", but utilizing the PTC* to fast construct PointELMs (PointELM-PCT*), the performance of the PCT* on the "Split-2" and "Split-3" improves largely.

E. Investigation of random initialization

For an untrained backbone network, initializations may impact the performance of PointELM. We conduct experiments on the Furniture dataset [22] and investigate serval random initializations. As shown in Fig. 6, except for the uniform initialization, other manners exhibit comparable performance on detailed accuracy evaulation. The Kaiming uniform and

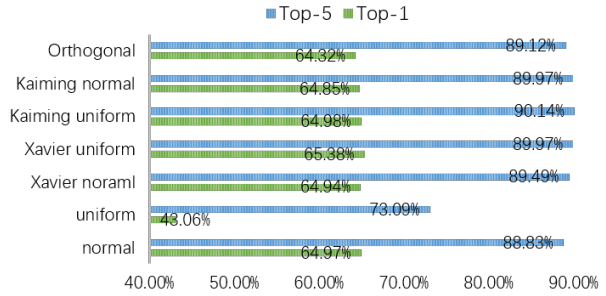

Fig. 6: Investigating different random initialization methods.

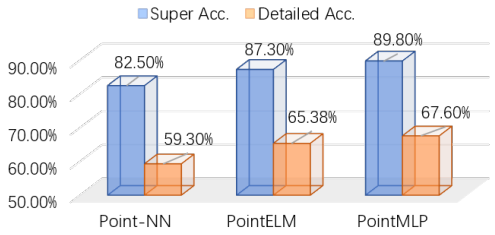

Fig. 7: Comparing classification accuracy on Furniture [22].

Xavier uniform show superior results in Top-5 and Top-1 accuracy, respectively, but not much more than the default Kaiming normal. In Fig. 7, a comparison of super and detailed accuracy demonstrates that PointELM approaches the performance of PointMLP and notably surpasses that of Point-NN. It reflects PointELM bridges the gap between Point-NN and conventional point cloud networks.

IV. Conclusion and Future Work

In this study, we propose PointELM, showcasing that point cloud classification doesn't rely on training a deep network. It leverages a randomly-weighted network as the backbone for extracting infant features, followed by a frequency-domain mapping to enhance these features. Finally, an ELM classifier is employed to categorize the enhanced features. Owing to training backbone is unnecessary, PointELM exhibits an exceptional learning speed. While its accuracy only decreases by 2.8% compared to a trained DGCNN, it trains approximately $200\times$ faster than the DGCNN. Extensive experiments highlight PointELM's adaptability to dataset variations and point cloud sampling. Moreover, PointELM evidences that a randomly-weighted network has commendable feature extraction capabilities for handling point cloud, presenting a fresh perspective for subsequent 3D research endeavors. Our future work will delve deeper into exploring PointELM's application in other tasks, such as segmentation, and consider its integration into point-based large language models. Finally, we conjecture deep networks may not be necessarily trained, and designing a network with appropriate random weights is also a rational way to implement AI.

REFERENCES

[1] Yuchao Zheng, Yujie Li, Shuo Yang, and Huimin Lu, "Global-pbnet: A novel point cloud registration for autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 11, pp. 22312–22319, 2022.

[2] 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, pp. 77–85, IEEE Computer Society.

[3] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *NeurIPS*, 2017, pp. 5099–5108.

[4] Mingqiang Wei, Zeyong Wei, Haoran Zhou, Fei Hu, and Huajian Si et al., "Agconv: Adaptive graph convolution on 3d point clouds," *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 1–18, 2023.

[5] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, and Michael M. Bronstein et al., "Dynamic graph CNN for learning on point clouds," *ACM Trans. Graph.*, vol. 38, no. 5, pp. 146:1–146:12, 2019.

[6] Meng-Hao Guo, Junxiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R. Martin, and Shi-Min Hu, "PCT: point cloud transformer," *Comput. Vis. Media*, vol. 7, no. 2, pp. 187–199, 2021.

[7] Guohao Li, Matthias Müller, Ali K. Thabet, and Bernard Ghanem, "Deepgcns: Can gcns go as deep as cnns?," in *ICCV*. 2019, pp. 9266–9275, IEEE.

[8] Xu Ma, Can Qin, Haoxuan You, Haoxi Ran, and Yun Fu, "Rethinking network design and local geometry in point cloud: A simple residual MLP framework," in *ICLR*. 2022, OpenReview.net.

[9] Nan Zhang, Zhiyi Pan, Thomas H. Li, Wei Gao, and Ge Li, "Improving graph representation for point cloud segmentation via attentive filtering," in *CVPR*. 2023, pp. 1244–1254, IEEE.

[10] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Hammoud, Mohamed Elhoseiny, and Bernard Ghanem, "Pointnext: Revisiting pointnet++ with improved training and scaling strategies," in *NeurIPS*, 2022.

[11] Renrui Zhang, Liuhui Wang, Yali Wang, Peng Gao, Hongsheng Li, and Jianbo Shi, "Starting from non-parametric networks for 3d point cloud analysis," in *CVPR*. 2023, pp. 5344–5353, IEEE.

[12] Zhuangzi Li, Ge Li, Thomas H. Li, Shan Liu, and Wei Gao, "Semantic point cloud upsampling," *IEEE Trans. Multim.*, vol. 25, pp. 3432–3442, 2023.

[13] Guang-Bin Huang, "What are extreme learning machines? filling the gap between frank rosenblatt's dream and john von neumann's puzzle," *Cognitive Computation*, vol. 7, no. 3, pp. 263–278, 2015.

[14] Laurens van der Maaten, "Accelerating t-sne using tree-based algorithms," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3221–3245, 2014.

[15] Zhuangzi Li, Xiaobin Zhu, Lei Wang, and Peiyu Guo, "Image classification using convolutional neural networks and kernel extreme learning machines," in *IEEE International Conference on Image Processing, ICIP 2018*. 2018, pp. 3009–3013, IEEE.

[16] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, and Nithin Raghavan et al., "Fourier features let networks learn high frequency functions in low dimensional domains," in *NeurIPS*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, Eds., 2020.

[17] Chiheb Trabelsi, Olexa Bilaniuk, Ying Zhang, Dmitriy Serdyuk, Sandeep Subramanian, João Felipe Santos, and et al., "Deep complex networks," in *ICLR*. 2018, OpenReview.net.

[18] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar, *Foundations of Machine Learning*, MIT Press, 2012.

[19] 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, pp. 1912–1920.

[20] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung, "Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data," in *ICCV*, 2019, pp. 1588–1597.

[21] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, and Zimo Li et al., "Shapenet: An information-rich 3d model repository," *CoRR*, vol. abs/1512.03012, 2015.

[22] Huan Fu, Rongfei Jia, Lin Gao, Mingming Gong, Binqiang Zhao, Stephen J. Maybank, and Dacheng Tao, "3d-future: 3d furniture shape with texture," *Int. J. Comput. Vis.*, vol. 129, no. 12, pp. 3313–3337, 2021.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *ICCV*. 2015, pp. 1026–1034, IEEE Computer Society.

[24] Guang-Bin Huang, Zuo Bai, Liyanaarachchi Lekamalage Chamara Kasun, and Chi-Man Vong, "Local receptive fields based extreme learning machine," *IEEE Comput. Intell. Mag.*, vol. 10, no. 2, pp. 18–29, 2015.

[25] Itai Lang, Asaf Manor, and Shai Avidan, "Samplenet: Differentiable point cloud sampling," in *CVPR*. 2020, pp. 7575–7585, IEEE.