Time and Memory Efficient 3D Point Cloud Classification

Shan Ullah
School of Electrical Engineering and Computer Science
National University of Sciences and Technology
Islamabad, Pakistan
Email: 11mseesullah@seecs.edu.pk

Usman Qayyum
Centers of Excellence in Science and Applied Technologies
Islamabad, Pakistan
Email: mrusmanqayyum@gmail.com

Aadil Jaleel Choudhry
School of Electrical Engineering and Computer Science
National University of Sciences and Technology
Islamabad, Pakistan
Email: 11beeachoudhry@seecs.edu.pk

***Abstract*—Recent advancement in real-time 3D scanning has upraised the need to maximally crunch the 3D data in minimum possible time. PointNet is a modern end-to-end deep learning architecture which directly utilizes point cloud without transforming it to regular formats for object- classification, part segmentation and scene semantic parsing. In this paper, we have proposed a novel set of hyper-parameters for PointNet which significantly reduced the size of network while achieving the same accuracy in 3D object classification as of the original network. This can enable the PointNet to be deployed on less resourceful hardware. The modified PointNet utilizes 16.5 Million less parameters (weights) and 5.2 Million lesser memory units. Moreover, through ample experiments, we demonstrate that our proposed configuration is highly efficient in terms of training and testing time.**

***Index Terms*—point cloud, 3D object classification, end-to-end deep learning**

I. Introduction

Existing research on end-to-end deep learning techniques for 3D object classification is primarily focused on achieving higher accuracy at the cost of computational resources. In this regard, many innovative deep learning architecture for various 3D representations have been proposed over the past few years which include Voxnet [2], [3], [4] for Voxels grids, Multi-view [4], [5] for set of images of 3D structure and PointNet [1] for point clouds. In parallel, researchers have endeavored to reduce the size and computations of a variety of deep learning architectures. Point cloud is a geometric structure comprising of unordered set of data in an irregular format. To the best of our knowledge, PointNet is a pioneer and state-of-the-art end-to-end deep learning architecture which consumes point cloud directly for 3D object classification and segmentation without converting it in any regular format such as voxel grids, meshes or set of image, etc. PointNet obtains high accuracy for 3D object classification and segmentation at the cost of high computational power and resources. This research work focuses on classification part of PointNet and proposed new set of parameters which reduces the size of network while achieving the same performance as of original. This enables the Point-Net to be deployed in those platforms having less memory and resources. We have named our proposed size of Point-Net as Shallow Point-Net architecture. In this fast-growing era where advancement in real-time 3D scanning technology is at peak, conversely, many cost-effective hand-held hardware has lesser memory. There is need to make compact architecture which could be utilized by maximum users. Our proposed size of PointNet is in much curtailed form which can replicate the original results utilizing less resources. Point-Net classification network aggregate global point cloud feature that can be used for object classification. Point-Net model is evaluated on ModelNet40 shape classification benchmark. Our proposed shallow Point-Net utilizes almost 16.5 Million less parameters and 5.2Million lesser Memory size(depends on memory unit) for Batch size of 32 than the original Point-Net [1] while achieved test set evaluation accuracy same as of original Point-Net for 3D object classification.

The main contribution of this research is:

- reproducing accuracy of original PointNet with 16.5 million lesser parameters and 5.2 million lesser memory units.
- reduction of training time by 1000 sec for 250 epochs on ModelNet40 dataset.

The proposed technique also reduces overfitting in original PointNet.

The paper is organized as follows: in Section II, we review and analyze PointNet as our benchmark for 3D object classification; we have explored current research in Section III for feasibility of applying existing techniques for reduction of PointNet; the proposed methodology is discussed in Section IV while results and conclusion are discussed in Sections V and VI, respectively.

II. Case Study : PointNet

PointNet [1] is a deep learning architecture that directly takes point clouds as input and outputs class labels for 3D Object classification. For algorithm validation, a freely available dataset known as 'ModelNet40' [3] has been used which contains CAD models consist in a 40 categories. Each 3D object is represented by a total of 2048 points. The original paper used uniformly sampled 1024 substantial points out of 2048 which could represent the shape of each object from all 40 categories.

978-1-5386-7729-2/19/$31.00©2019 IEEE

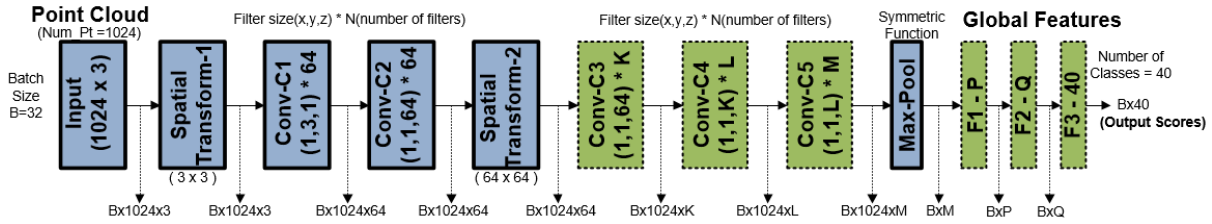

Fig. 1. PointNet [1] architecture for 3D object classification, where Blue boxes (with solid-line) are highly sensitive parts of network while Green boxes (with dotted-line) are less sensitive parts of network.

While training the network, point cloud is treated with random rotations in up-axis and with zero mean Gaussian noise jitter with 0.02 standard deviation. Full classification network of PointNet can be seen in Figure 1. PointNet has three key modules: (i) Spatial Transformations (ST-1 and ST-2) that align both input points and point features; (ii) the max pooling layer as a symmetric function to aggregate information from all the points and (iii) global features combination structure. Empirically, it has been observed that among these three key modules of PointNet, (i) and (ii) are more sensitive parts of network as disturbing these modules can produce large amount of degradation in performance of PointNet, while module (iii) is comparatively less sensitive part of network in terms of configuration changes. The first spatial transformation (ST-1) in Figure 1 predicts an affine transformation matrix (3x3) to align each set of raw input point clouds (1024x3) while second transformation (ST-2) predicts a feature transformation matrix (64x64) to align features from different point clouds. These spatial transformations are fundamental parts of PointNet and comprise multi convolutional layers of depth 64, 128 and 1024; a max-pooling layer and two fully connected layers with output sizes of 512 and 256. This enables learned features to be invariant to geometric transformations in point cloud. ST-2 along with some regularization term added to softmax training loss(L_{ref}) is used for optimization in [1] as:

$$L_{reg} = ||I - AA^T||_F^2 \tag{1}$$

One of the contributions of PointNet is its symmetric function for unordered points which is applied on transformed elements in the set:

$$f(x_1, ..., x_n) \approx g(h(x_1), ..., h(x_n)), \tag{2}$$

where $f : 2^{\mathbb{R}^N} \rightarrow \mathbb{R}$, $h : \mathbb{R}^N \rightarrow \mathbb{R}^K$ and $g : \mathbb{R}^K \times .. \times \mathbb{R}^K \rightarrow \mathbb{R}$ is a symmetric function

PointNet approximates h by a multi-convolutional layer network and g by a composition of a single variable and a max pooling function. Qi *et al.* [1] write that f can be arbitrarily estimated by PointNet provided with enough number of neurons at the max pooling layer i.e, K in Eqn. 2 is sufficiently large. This implies that size of max pooling layer is a sensitive parameter in terms of PointNet performance. From here, we can conclude that alignment networks (input and feature transformations) and symmetric function(max pooling layer) are most sensitive parts of the network. This conclusion shall form the basis of our methodology in Section IV.

III. Related Work

Over the years, researchers have proposed numerous state-of-the-art methods to reduce the size of end-to-end deep learning architectures. These include network pruning [11], [8], quantization (reduction in number of bits), weight sharing, Huffman coding [8] and Filter decomposition [6], [7] to design smaller network and have been successfully applied to famous deep learning architectures. Recently, network aware and adaptive methods [9], [10] for reducing number of MACs (multipliers and accumulators) have been proposed. We explored feasibility of applying aforementioned techniques on PointNet. Details are as following:.

- Network Pruning:
  This method prunes redundant connections by learning only the substantial ones [11], [8]. It begins with a pre-trained model, where the connections below certain threshold are removed by replacing those parameters with zeros. This type of model compression generates sparsity in network. PointNet is already dealing with sparse set of point cloud using its alignment networks and symmetric function. The sparsity produced in model using network pruning can further burden PointNet and hence it is not intuitive to directly apply network pruning and it may rather require additional spatial transformations. Therefore, in this work, network pruning has not been applied and has been left for future work.
- Parameter Quantization:
  Reducing number of bits to represent weights can further compress the size of network which in turn can boost efficiency. However, some quantization schemes are not a generalized solution for all processors to fully utilize bits reduction and speed-up the network. An example of this includes Deep Compression [8] which uses a codebook as part of its scheme for quantization of parameters (6 or 8 bits). It is hard to achieve speed-up of 4x (32/8) with 8-bit or 5.3x (32/6) with 6-bit quantization. In this regard, authors of [8] proposed special hardware to fully exploit the scheme. This drawback was highlighted by [7] with some common processors. In case of PointNet, as each point is treated independently for spatial transformation, introduction of quantization may disturb the alignment estimation for both input and feature transformations of point cloud. It is not trivial to apply quantization and is left for future work.

TABLE I
NUMBER OF PARAMETERS AND MEMORY CALCULATIONS FOR POINTNET - ORIGINAL AND PROPOSED

Batch Size - "B"		Input	ST-1	C1	C2	ST-2	C3	C4	C5	Pool	F1	F2	F3	Total	Diff
Memory (B=32)	Original	98304	39879424	2097152	2097152	39879424	2097152	4194304	33554432	32768	512	256	40	123930920	**5.2M**
	Proposed	98304	39879424	2097152	2097152	39879424	524288	524288	33554432	32768	16	64	40	118687352	
Parameters (B=32)	Original	0	17050057	192	4096	18100288	4096	8192	131072	0	16777216	131072	10240	52216521	**16.5M**
	Proposed	0	17050057	192	4096	18100288	1024	256	16384	0	524288	1024	2560	35700169	
Memory (B=1)	Original	3072	1246976	65536	65536	1246976	65536	131072	1048576	1024	512	256	40	3875112	**0.16M**
	Proposed	3072	1246976	65536	65536	1246976	16384	16384	1048576	1024	16	64	40	3710584	
Parameters (B=1)	Original	0	797129	192	4096	1847360	4096	8192	131072	0	524288	131072	10240	3457737	**0.77M**
	Proposed	0	797129	192	4096	1847360	1024	256	16384	0	16384	1024	2560	2686409	

- Number of Layers:
  It has been shown in literature that depth of network (no. of layers) plays an important role in the accuracy of any deep learning architecture. VGG [12] proposed CNNs with 12 to 19 layers and reported that deeper networks produce higher accuracy. Similarly [13] shows that going deeper improves accuracy. In [14], CNNs of up to 30 layers have been proposed for achieving higher accuracy. Furthermore, alteration in number of layers leads to major architectural change and can disturb convergence of networks. Empirically, it has been observed that increasing layers in original PointNet has no significant effect on accuracy. However decreasing the number of layers slightly decreases the accuracy as well. Therefore, to reduce number of parameters, it is not trivial to perform major architectural change. Nevertheless, modification in number of layers will be considered for future work.
- Filter Decomposition and Depth of Layer:
  In recent work [6], [7], the receptive field is reduced by replacing the standard convolutional filters with two layers such as (a) depth-wise convolution combined with point-wise convolution (1x1) to build depth-wise separable filters. This technique [7] has revolutionized the usage of filters in deep learning architectures and has achieved much success by using 50x fewer parameters to achieve AlexNet [15] accuracy. PointNet is already exploiting point-wise (1x1) filters to extract global feature as shown in Figure 1 and claims to be efficient in time and space complexity as compared to existing techniques [5]. Therefore, it is difficult to further exploit the x and y-dimensional filtering. However, depth of filter and depth of particular convolutional layer has been utilized in our proposed work.

IV. METHODOLOGY

The primary goal of our work is to propose time and memory efficient 3D point cloud classification. In this regard, after study and analysis for feasibility of aforementioned state-of-the-art pruning techniques on PointNet, it can be concluded that reducing depth of filters for each convolution layer in less sensitive part (shown in Figure 1 with dotted-line) of network is more trivial. Initially starting from left most block of network shown in Figure 1, number of points of point cloud to represent one particular 3D shape is 1024. Empirically, the behavior of network was analyzed by reducing points to 512 and then to 256. Degradation in performance was observed. Similarly, second block from left is spatial transformation-1 (3x3 matrix) for alignment of input point cloud which in turn consists of convolutional (64,128 and 1024) layers, a max-pooling layer and two fully connected (512,256) layers. We varied the number of layers as well as depth of each layer(conv and FC) for Spatial Transformation (ST-1) and observed decline in performance. Extraction of aligned input point cloud is performed via two convolutional layers (C-1 and C-2) followed by Spatial Transformation(ST-2). Therefore, this particular transformation is also known as feature transformation (64x64 matrix). Up to this point in network, varying hyper-parameters results in significant decline in performance of PointNet, thereby categorizing aforementioned layers as sensitive.

After feature alignment, three convolutional layers (C3 to C5) are used to extract global features using depth of K, L and M, originally set to 64, 128 and 1024. As depicted in Figure 1, number of filters in one layer acts as the depth of filter in the subsequent layer. Empirically, we inferred from our analysis that depth of each layer (C3 to C5) has less significant effect on performance of PointNet. Some of the variations in K, L and M with shallow configuration are shown in Table III. Besides this, we also increased the depth of layers (C3 to C5) and found no major difference in performance of PointNet.

Another key module of PointNet is its symmetric function (max pooling layer) which receives input of size Bx1024xM and outputs BxM where B is batch size and M is the depth of C5. Empirically, we found that in order to achieve perfect symmetry, size of C5 should be equal to number of points.For instance, max pooling layer receives input of size Bx1024x1024 and outputs Bx1024. To retain perfect symmetry, we propose to keep the size of C5 equal to the input. Up to this point in the network, final set of proposed hyper-parameters for layers C3 C4 and C5 would be K=16, L=16 and M= 1024 (equal to number of input points) respectively, as shown in Table III. In addition, a theorem in [1] states that the size of max pooling layer should be large for achieving favorable symmetry. Therefore as number of input points is 1024, combining both strategies, size of C5 in proposed configuration remains the same as that of PointNet, as shown in Table III.

Max pooling layer is followed by the remaining part of global feature extraction for object classification which con-

TABLE II
SUMMARY OF RESULTS

Batch size=32, Epoch=1	Original	Proposed	Diff
# of Params	52.2M	35.7M	**16.5M**
Memory	123.9M	118.7M	**5.2M**
Time per Train Example	82sec	78sec	**4sec**
Time per Evaluation	19.5sec	18.1sec	**1.4sec**

sists of fully connected layers(F1, F2 and F3) as shown in Figure 1. We varied the depth of F1 and F2 regardless of depth of F3 which is equal to number of classes/categories (40 in this case), and observed little to no effect on performance of PointNet. It can be thus inferred that aside from ST-1, ST-2 and Max-Pooling layer, size of PointNet can be significantly reduced without much affect on its accuracy and final set of proposed hyper parameters for Figure 1 are hereby given as K=16, L=16, M=1024 (or number of input points), P=16 and Q=64. With our proposed settings we have preserved evaluation accuracy of PointNet with far less parameters and memory as conferred in Section V.

V. RESULTS AND DISCUSSION

The proposed technique was evaluated on ModelNet40 shape classification benchmark. For fair comparison, all of the default settings were kept the same as in original paper and tensorflow code available on Github. PointNet was trained for a total of 250 epochs. Dataset was further divided into five separate blocks. Processing of one epoch contains training example randomly picked from each of the five blocks. In order to avoid overfitting, the five trained examples, one from each blocks, are evaluated at end of each epoch.

Although, PointNet paper claims an accuracy of 89.2% in object classification, Github code achieves a maximum of 88.7% using default settings on different computers and GPUs. Therefore for our work , benchmark is 88.7%. We have achieved the using 16.5 million lesser parameters and 5.2 million lesser memory units than original. Similarly, time to train one example by original PointNet is 82sec using GPU GeForce GTX-745 while the proposed technique uses 78sec to process same data. Summary of results is shown in Table II.

It can be seen from Figure 2 that evaluation accuracy after each epoch on test dataset by proposed technique and original settings is getting closer and then equal to 88.7% at the end of 250 epochs as the network converges. Although, training accuracy of original PointNet is 94% after 250 epoch, the proposed configuration achieves the training accuracy of 85%. This might be due to overfitting of original PoinNet which has been overcome by proposed technique. Similarly, difference in evaluation mean loss of original(0.51) and proposed(0.58) network is smaller as compared to difference of training mean loss of both.

Parameter calculations for both original PointNet and proposed configuration can be seen at Table I which depict that proposed technique utilizes 16.5 Million lesser parameters and 5.2 Million lesser memory unit than that of original PointNet.

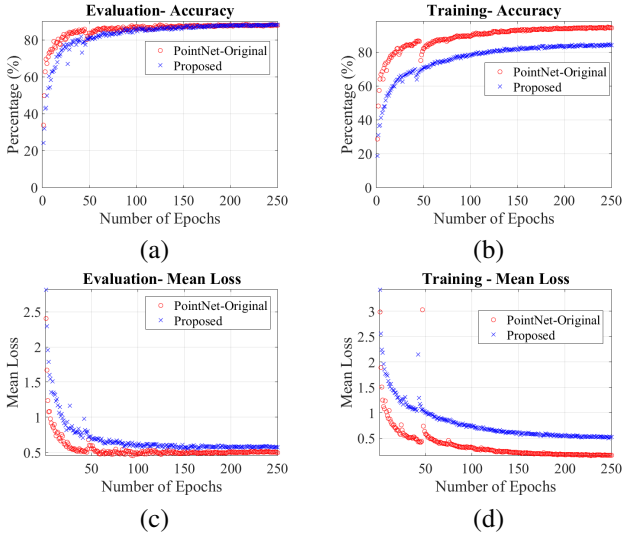

Fig. 2. Accuracy and mean loss curves for training and evaluation of PointNet and proposed work.

TABLE III
EVALUATION ACCURACY AND MEAN LOSS FOR SET OF COMBINATION OF DEPTH (K,L,M,P AND Q) OF FILTERS."C" STANDS FOR CONVOLUTIONAL LAYER AND 'F' STANDS FOR FULLY CONNECTED LAYER.

	K-C3	L-C4	M-C5	P-F1	Q-F2	Loss	Acc
Original	64	128	1024	512	256	0.51	0.887
Proposed	16	16	1024	16	64	0.58	0.887
Shallow1	32	64	512	512	128	0.57	0.873
Shallow2	32	64	256	256	128	0.44	0.879
Shallow3	32	64	128	128	64	0.63	0.872
Shallow4	32	64	64	64	64	0.53	0.878
Shallow5	32	32	32	32	64	0.59	0.871
Shallow6	32	16	16	16	32	0.57	0.857
Shallow7	16	16	16	16	64	0.59	0.857
Shallow8	16	8	8	8	8	0.94	0.791
Shallow9	16	8	8	8	16	1.18	0.698

As shown in Table II, proposed method processes one training example in 78 sec and faster by 4sec than original PointNet on GeForce GTX-745. In case of 250 epochs, proposed technique finishes training 1000sec (16.66 minutes) prior to original PointNet. However, for more complex datasets of point cloud, where more number of epochs are required to converge network, propose technique will converged well before the original network.

VI. CONCLUSION

The paper presents a humble effort to optimize PointNet in terms of number of parameters, memory and time. The paper categorizes significance and sensitivity of each part of PointNet and identifies parts after feature transformations except symmetric function as potential reduction areas. After discussing state-of-the-art pruning techniques and their intuitive detrimental effects on PointNet, the paper proposes decreasing depth of layers in potential reduction areas. The paper also proposes to maintain max pooling layer size equal to input point cloud. Through this, the research has achieved

reproducing accuracy of original PointNet with 16.5 million lesser parameters and 5.2 million lesser memory units. A reduction of training time by 1000 sec for 250 epochs on ModelNet40 dataset has also been reported. The paper also discusses potential decrease in overfitting as compared to original PointNet.

Original PointNet paper presents results for segmentation of point clouds as well. Therefore, future works includes comparison of segmentation accuracy of original PointNet with proposed configuration. Authors are also exploring the possibility of using hyper-columns in order to achieve accuracy higher than that of original PointNet while using lesser parameters.

VII. Acknowledgment

Authors would like to thank Charles Ruizhongtai Qi *et al.* for making their PointNet [1] code available as open source on Github. It not only accelerated our efforts in reproducing their published results as benchmark but the proposed optimizations were also made possible by directly exploring their code.

References

[1] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, vol. 1, no. 2, p. 4, 2017.

[2] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 922–928.

[3] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1912–1920.

[4] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view cnns for object classification on 3d data," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5648–5656.

[5] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 945–953.

[6] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[7] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[8] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[9] Q. Huang, K. Zhou, S. You, and U. Neumann, "Learning to prune filters in convolutional neural networks," *arXiv preprint arXiv:1801.07365*, 2018.

[10] T.-J. Yang, A. Howard, B. Chen, X. Zhang, A. Go, M. Sandler, V. Sze, and H. Adam, "Netadapt: Platform-aware neural network adaptation for mobile applications," *Energy*, vol. 41, p. 46, 2018.

[11] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in neural information processing systems*, 2015, pp. 1135–1143.

[12] K. S. . A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[13] W. L. R. e. Christian Szegedy, "Going deeper with convolutions," 2014.

[14] S. R. K. He, X. Zhang and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," 2015.

[15] I. S. Alex Krizhevsky and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," 2012.