Fuzzy Lightweight CNN for Point Cloud Object Classification based on Voxel

1st Oddy Virgantara Putra
Department of Electrical Engineering
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
7022221024@mhs.its.ac.id

2nd Moch. Iskandar Riansyah
Department of Electrical Engineering
Institut Teknologi Telkom Surabaya
Surabaya, Indonesia
iskandar@ittelkom-sby.ac.id

3rd Riandini
Department of Electrical Engineering
Politeknik Negeri Jakarta
Depok, Indonesia
riandini@elektro.pnj.ac.id

4th Ardyono Priyadi
Department of Electrical Engineering
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
priyadi@ee.its.ac.id

5th Eko Mulyanto Yuniarno
Department of Electrical Engineering
Department of Computer Engineering
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
ekomulyanto@ee.its.ac.id

6th Mauridhi Hery Purnomo
Department of Electrical Engineering
Department of Computer Engineering
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
hery@ee.its.ac.id

***Abstract*—Point cloud object classification has gained attention from many researchers since the emergence of public dataset like ModelNet and ShapeNet, which contains full surface objects. However, in practice, objects captured using LiDAR are only partially covered in the scanned area, making such a task burdensome. Here, we proposed a solution to overcome those problems. It is a novel fuzzy convolutional inference (FuzzConv) incorporated with depthwise over-parameterization (DOConv). Instead of applying raw data, the point clouds are transformed into a 3D voxel. We utilized EfficientNet as our backbone and modified the Mobile inverted Bottleneck Convolution (MBConv) with DOConv. In the last fully connected (FC) layer, we added the FuzzConv layer as an inference before feeding the feature map to the output layer. Consequently, to validate the performance of our model, we undertake an evaluation with multiple classifications in ModelNet10, ModelNet40, and our core dataset, the point cloud of human poses. Accuracy, loss, number of parameters, loss, precision, and F1-scores are employed as performance indicators. As a result, our model achieved top performance regarding the accuracy and loss value for the primary dataset, 83 % and 0.56, for ModelNet10 88.1 % and 0.56, and ModelNet40 74.1% and 1.15.**

***Index Terms*—Fuzzy Convolution, Lightweight CNN, Human Pose, Point Cloud Classification, Voxel**

I. Introduction

Nowadays, the development of LiDAR technology has been advanced in many areas. This advancement envelops researches in automatic vehicle [1]–[3], remote sensing [4], and human activities [5], [6]. Thanks to this cutting-edge technology, we can acquire large 3D data at a reasonable cost [7].

Several formats, for example, point clouds (PC), 2.5-D images, and volumetric shapes can typically be treated to represent 3D data. PC representation keeps the initial geometric data in the euclidean domain without quantization. Thus, representing such data is challenging, especially in self-driving cars and humanoid robots. Until now, researchers in deep learning, particularly in 3D point clouds, faced incredible obstacles due to its unstructured nature and high dimensionality [8]. By the existence of the public dataset, such as KITTI [9], ModelNet10, ModelNet40 [10], and ShapeNet [11], the research regarding point cloud propelled up high. It increased the number of approaches being put forth to solve a variety of point cloud processing-related issues, such as 3D PC classification, detection, tracking, segmentation, registration, and reconstruction.

Our paper's main contribution is a novel lightweight CNN model by modifying the convolution layer incorporating a FuzzConv. Furthermore, our model achieves the best results in size with small parameters while improving accuracy.

The remaining sections are organized as follows: Section II, Related Works, Section III, Proposed work, discusses our model. Section IV, Experiment and Result, elaborates on verification and classification performance, and Section V is Conclusion.

II. Related Works

1) Direct Point-wise methods: Directly processing raw point clouds prevalently into 2D deep learning is impossible due to its fickle nature. Charles [12] proposed a deep learning network to extract 3D geometric from point clouds called PointNet. This model directly feeds the raw point cloud to the network. Instead of putting all points, this method samples data with only a 2048 sample set. To be precise, PointNet utilizes several layers of MLP to classify objects based on pointwise features. Deepsets [13] accomplish permutation invariance by aggregating all points of nonlinear transforms. Wang [14] proposed a classification model based on a point cloud graph. The feature model is learned from space and updated sequentially on each layer. The inner core of the model is EdgeConv and MLP. In EdgeConv, the points are aggregated using a channel-wise operator. Another paper [15] proposed a network based on adaptive feature adjustment for point cloud recognition. It utilized fully connected point pairs within regions. Momenet [16] took advantage of geometric moments of point cloud to classify shapes. A multiple levels contextual encoding technique for point categorization has been proposed by [17]. By structurally considering a point and its surrounds, we aim to address a common and generic feature learning challenge in 3-D PC classification: how to express geometric characteristics more effectively and discriminatively. Liu [18] came up with a context-aware network.

2) Convolution-based methods: The raw point cloud is burdensome to study due to its irregular shape. Several projects have been pitched to address this problem.. A geometry topology relation-based neural network (RS-CNN) [19] is proposed to handle the uncertainty in the underlying shape of point clouds. This work is tested on ModelNet40, which contains full-body 3D object point clouds. However, this method is prone to failure if opposed to partially available 3D object PCs. In the following year, a 3D fractal-based voxel transformation [20] is proposed to tackle an issue about the real-life dataset and continued by [21] using fractal keypoint detector. Even though 3D transformer-based learning considerably improves point cloud analysis, it exposes to extensive computational cost [22].

III. Proposed Work

In this work, we proposed a framework for object classification based on voxel. This framework incorporated a deep learning model as our backbone called EfficientNet, a fuzzy inference layer (FL) [23], and depthwise over-parameterization (DO) [24]. In the backbone, we modified several layers to add FL and DO. The whole proposed work is displayed as in Fig. 1.

A. Point Cloud to Voxel

Here, we utilize a primary dataset (D1) containing human pose and a secondary dataset from ModelNet10 and ModelNet40. ModelNet10 contains ten classes of point cloud objects. Meanwhile, ModelNet40 contains 40 classes. Our primary dataset is acquired using a 32-channel LiDAR sensor in a 30-meter square room. LiDAR sensor records the data in Packet Capture (PCAP) format, a sequential point cloud data (PCD). In our D1, since we only need the 3D coordinate, we extracted the points from PCAP into PCD format. Then, for the ModelNet10 and ModelNet40, we parsed them to PCD from mesh. Prior to voxelization, we normalized the 3D coordinate to standardize and reduce computational performance. The voxel size is set to 16 for each axis. However, instead of directly putting the number of PCs into the voxel, we determine the value of the voxel based on the existence of points. The rule as in Equation 1:

$$V(x) = \begin{cases} 1, & \text{if } P \geq 1 \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where V is the voxel value, x represents XYZ-coordinates, P is the number of point cloud within the voxel. From this process, we have voxels with the size of $(16, 16, 16)$.

B. Oversampling

Data augmentation is one technique to tackle the issue of data imbalance. In this area, there are two terminologies: under-sampling and over-sampling. Under-sampling is a method to reduce the amount of data in the majority classes to match the minority. The benefit of this method is time efficiency and low computational process. However, the shortcomings of this method are that some vital information is lost due to data reduction. For its counterpart, it is defined as a data replication of the minority class to match the majority. By using over-sampling, the classification performance might increase. However, it might lead to over-fitting due to replicative data. Thus, we employ Synthetic Minority Oversampling Technique (SMOTE). SMOTE augments by generating artificial data points based on the original data points. SMOTE has the benefit of not producing duplicate data points but instead producing artificial data points that are marginally different from the actual data points. Since all of datasets are imbalanced, thus we implement SMOTE.

C. The Backbone

The backbone of our proposed framework works as the primary deep learning model. When selecting this model, we evaluated several classifiers algorithms such as ResNet, EfficientNet, MobileNetV2, VGGNet, and PointNet. Among those models, we picked a model with high performance in terms of accuracy and the number of parameters. Thus, we found out that EfficientNet worked as our backbone.

Our backbone contains three batches. A single batch contains a 2D-Convolutional Layer (Conv2D), batch normalization (BN), Leaky ReLU (LR), and two Depthwise Convolution (DC). We use Depthwise Over-parameterization (DO) as a convolutional layer instead of regular Conv2D. In the next layer, BN and LR are the activation functions. For the next layer, we have DC and max pooling.

D. Depthwise Over-parameterization Layer

Depthwise Over-parameterization Convolution (DOConv) is a combination of DC and Conv2D. In addition to being demonstrated as a method for accelerating the linear network training process, it has also been empirically demonstrated to quicken the training of deep non-linear networks [24], [25]. These results imply that, despite the substantial effort put into the search for new network topologies, over-parameterization offers a significant untapped potential for improving existing structures. Therefore, instead of conventional Conv2D, we employ the Over-parameterization Conv2D layer.

A conventional Conv2D is denoted as multiplication between a sliding window P and its corresponding matrix with the size of p x l. Consider a 3D matrix W with the size of C_{in} x C_{out} x M x N, a sliding window P with the size of C_{in} x M x N. Thus, we have a feature map as in Equation 2

$$\mathrm{O} = \mathrm{W} * \mathrm{P} \tag{2}$$

where $(*)$ is a convolutional dot-product operator. The result extends on the size of feature map

On the other hand, DC is a depth convolution extending to a matrix depth. An input matrix with a certain number of channels (N_c) is separated into N_c batch(es). Each batch undergoes a convolutional operation. In the last process, the results were stacked. Thus, we have an output tensor. The number of dividers for each batch is also called D_{mul}, representing the feature dimension.

Consider a 3D matrix W with the size of (D_{mul}, C_{in} , M , N), a sliding window P with the size of C_{in} x M x N. Thus, we have a feature map as in Equation 3

$$\mathrm{O} = \mathrm{W} \circ \mathrm{P} \tag{3}$$

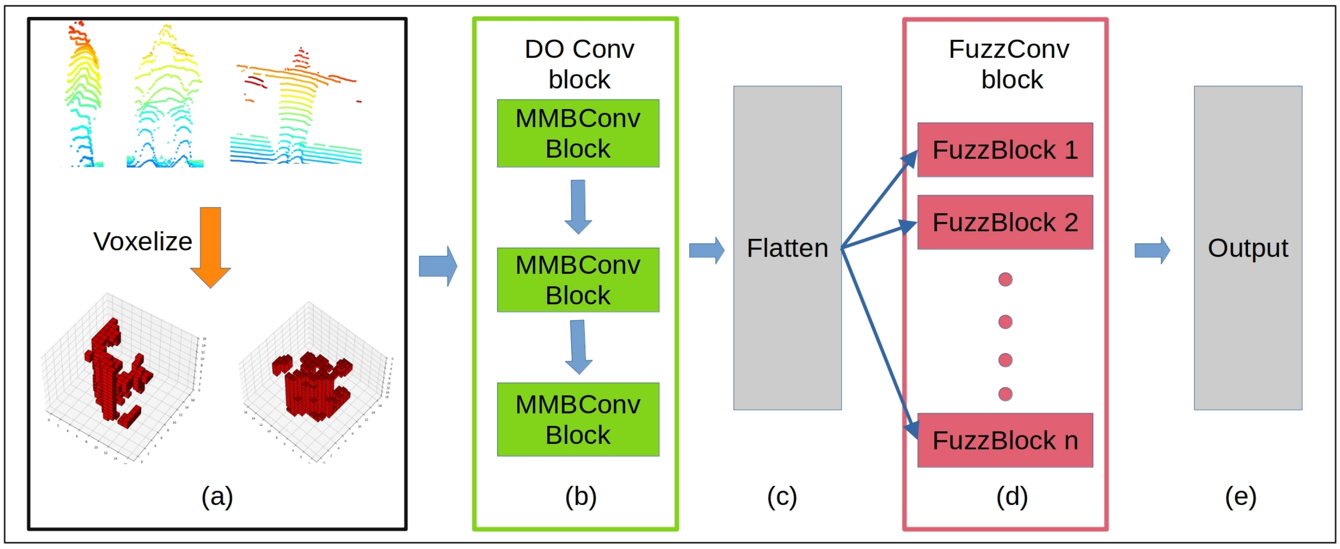

Fig. 1. Our proposed work. This work is composed of five blocks, (a) voxelization block, (b) DOConv block, (c) 1-Dimensional transformation, (d) FuzzConv block, and (e) is the output.

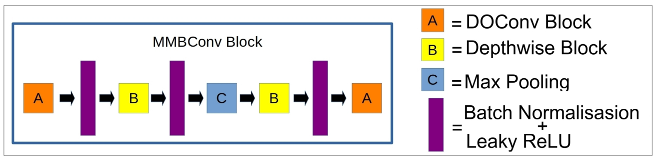

Fig. 2. Our modified MBConv (MMBConv) Block.

where ($\circ$) is a depthwise convolutional operator. Imagine a 3D matrix W with the first dimension (row) with C_{in}, the second dimension (column) with MxN, and the third dimension with D_{mul}. Then, W calculated with a sliding window P where P is composed of a 2-dimensional matrix with C_{in} and MxN in the 1st and 2nd dimension respectively. From depthwise operation, we get O.

A DOConv is composed from depthwise kernel $\mathbb{D} \in R^{(M\times N)\times D_{mul}\times C_{in}}$ and Conv2D with kernel $\mathbb{W} \in R^{C_{out}\times D_{mul}\times C_{in}}$. Consider a sliding window $\mathbb{P} \in R^{(M\times N)\times C_{in}}$. From $\mathbb{P}$, $\mathbb{W}$, and $\mathbb{D}$ we can see as in Equation 4:

$$\begin{aligned}\mathbb{O} &= (\mathbb{D}, \mathbb{W}) \circledast \mathbb{P} \\ &= \mathbb{W} * (\mathbb{D} \circ \mathbb{P}) \\ &= (\mathbb{D}^T \circ \mathbb{W}) * \mathbb{P}\end{aligned} \tag{4}$$

where $\mathbb{D}^T$ is derived from the first and second axis of $\mathbb{D} \in R^{M\times N} \times D_{mul} \times C_{in}$.

The DOConv is placed in convolution layer in the backbone which substitute the conventional one as seen on Fig. 2.

E. Fuzzy Inference Layer

Convolutional neural network (CNN) is considered a disruptive machine learning idea. It can extract features in the spatial domain. CNN works like a charm for reducing computational processes. Instead of reshaping a $m \times n$ 2D matrix into a 1D vector with a size of $1 \times (m \times n)$, CNN extracts only meaningful information from a matrix, thanks to the convolutional operator (CO). However, CO has its risks, which result in promoting a vanishing gradient.

Assuming there is a 3D matrix input x_i, $i = 0, 1, 2, ..., n$. We can formulate an output of extracted feature maps with Equation 5:

$$\begin{aligned}o_j^l &= \sum_{i=0}^{n}(x_i * w_{ij}^l) + b_j^l, j = 1, 2, ..., q^l \\ B &= f(o_j^l)\end{aligned} \tag{5}$$

where n is the number of channel, B is output feature maps after undergo an activation function f, w works as weight on channels i^{th} in a layer l, b is a bias, and lastly q is the number of output features.

In the fuzzy system, to generate fuzzy inference is based on fuzzy rules. For example, given inputs a_i, $i = 1, 2, ..., n$ and outputs $b_j, j = 1, 2, ..., m$, and the c^{th} fuzzy rule R^l is in Equation 6:

$$\begin{aligned}\text{IF } a_1 \text{ is } \mathbb{A}_1^l \; and \\ \text{IF } a_2 \; is \; \mathbb{A}_2^l \; and \\ ... \\ \text{IF } a_n \; is \; \mathbb{A}_n^l \\ \text{THEN } b_1 \; is \; w_1^l \; and \\ \text{THEN } b_2 \; is \; w_2^l \; and \\ ... \\ \text{THEN } b_n \; is \; w_n^l\end{aligned} \tag{6}$$

where $\mathbb{A}$ is the fuzzy set. The bigger fuzzy input implies arduous computation. Thanks to MISO Fuzzy, which helps to split the inputs, thus reducing the computational cost.

Occasionally, Conv2D works as a filter to extract features (fmaps) from input data. The fmaps are fed into the final layer, the fully-connected (FC) layer. For every single pixel in fmaps, it becomes crips values. For example, imagine we have three fuzzy sets $\mathbb{M} = 3$, "low, medium, and high" as MF if the number of fmaps from a Conv2D layer is k and

$k = 120$, where every feature map has a size of 3×3. Thus, we have a list of a fuzzy map with the size of $k \times \mathbb{M} = 120 \times 3 = 360$. However, the more extensive inputs, the higher the computational cost is. To tackle this issue, we cast a fuzzy neural network (FNN) to incorporate with our backbone. Instead of the FC layer, we utilize a semi-connected layer from FNN input to work as a feature map. Finally, the defuzzifier layer sums up the fuzzy machine.

In order to achieve the fuzzy inference, we can calculate using Equation 7:

$$\Theta_q^p = \prod_{i=1}^{h\times w} \mu_{F_i^q}(x_i) \tag{7}$$

where Θ_q^p resembles the set of output inferences q from feature maps p. As introduced in [23], the fuzzy set within the MF are $N, Z,$ and P. where:

$$N = e^{-\frac{(x+m)^2}{2\sigma^2}}, Z = e^{-\frac{x^2}{2\sigma^2}}, P = e^{-\frac{(x-m)^2}{2\sigma^2}} \tag{8}$$

For every feature map with a size of $n \times n$, it maps to $n \times n$ fuzzy maps as well as N, Z, and P. According to fuzzy set, the number of parameters in fuzzy is denoted as Equation 9:

$$\mathbb{N}^{k\times h\times w} \tag{9}$$

where k is the number of feature maps, w is the number of fuzzy set within MF. If we have a feature map with size $n = 3$, and $w = 120$, $\mathbb{N} = 3^{3\times3\times120}$. Seeing this, we adapt the reasonable $\mathbb{N}$ from [23] and split the feature map to independently execute the fuzzy inference. Thus, we have:

$$\Theta_q^p = \prod_{i=1}^{9} \mu_{F_i^q}(x_i) \tag{10}$$

where

$$\mu_{F_i^q}(\zeta_i) = \begin{bmatrix} N(\zeta_1) \times \ldots \times N(\zeta_9) \\ Z(\zeta_1) \times \ldots \times Z(\zeta_9) \\ P(\zeta_1) \times \ldots \times P(\zeta_9) \end{bmatrix} \tag{11}$$

where $\zeta_{p,i}^5$ represents pixels i after Conv2D layer in set p. Finally, with the number of fuzzy inference units R equals to $3^{3\times3} = 19683$, we can merge fuzzy layer with FC layer $z^8 = w^8 \times \psi$, where:

$$\psi = \begin{bmatrix} \Theta_1^1 \\ \Theta_1^2 \\ \vdots \\ \Theta_{19683}^{120} \end{bmatrix}, z^8 = \begin{bmatrix} z_1^1 \\ z_1^2 \\ z_{19683}^{120} \end{bmatrix} \tag{12}$$

and w^8 is defined as:

$$w^8 = \begin{bmatrix} w_{1,1}^8 & \cdots & w_{1,19683}^8 \\ w_{2,1}^8 & \cdots & w_{2,19683}^8 \\ w_{3,1}^8 & \cdots & w_{3,19683}^8 \end{bmatrix} \tag{13}$$

Consequently, the cost function used in here is:

$$f(x) = \frac{1}{e_{1^8}^z + e_{2^8}^z + e_{3^8}^z} [e_{1^8}^z e_{2^8}^z a] \tag{14}$$

In this fuzzy layer, we integrate Conv2D with Fuzzy Inference before feed the output the FC layer as seen as red block in Fig. 1 (d).

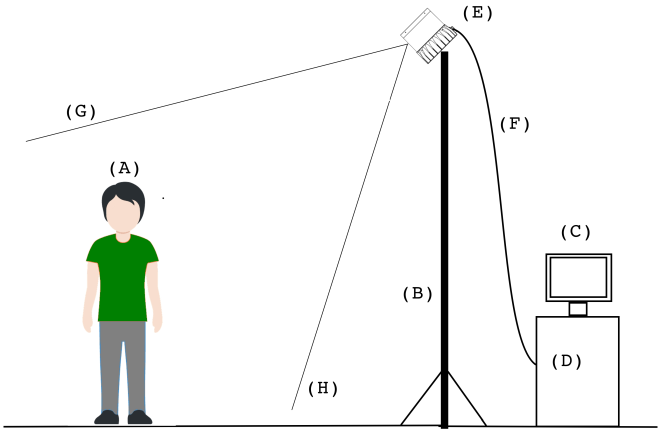

Fig. 3. Our Primary Data Acquisition Procedure. The subject (A) stands 3 meters from horizontal distance of the sensor position. Our laptop is connected to the interface box (D) which linked with cable (F) to the facing down LiDAR sensor (E)

F. Evaluation Method

We evaluated all classifiers to our model using classification metrics with several setups, the number of epoch, the optimizers, and the loss function.

IV. Experiments and Results

In here, we utilize a PC with processor Core i7, RAM 16GB, NVIDIA GTX 1050Ti for training our model. To verify our model performance, we evaluated and compared with other classifiers methods using two public datasets named ModelNet10 and ModelNet40 collected from [26].

A. Dataset Gathering

We gathered our primary dataset using a mid-range LiDAR OUSTER type OS1-32. The sensor is equipped with 32 channels, vertical field of view (FOV) 45°(up and down view 22.5°), precision up to 0.5 cm, and a range of 90 meters.

The subjects are four adults, 50 % male, with age around 34 years old. They performed poses such as hands up, crouching, sitting, squatting, standing, and lying down. They are required to stand in the center spot. Instead of aligning LiDAR with the subject, we mounted our sensor in a top corner of a room-sized 4x6 meter. The sensor is set to face 45 °down towards the center spot. This procedure is designed in such a way as to mimic surveillance in a small room.

We recorded each session using a LiDAR sensor with an area span between 90 and 270 degrees for 20 seconds. The outputs of the recorded files are in Packet Capture (PCAP) format. Finally, we extracted all raw point cloud data (PCD) from PCAPs.

ModelNet10 is a dataset consisting of PCD, which contains ten distinguished objects, while ModelNet40 consists of 40 objects. This PCD is constructed from a CAD model. To build a voxel, we create a voxel with a size of 16 for all axis. Thus, from raw PCD we get a 3D voxel with $(x, y, z) = (16, 16, 16)$. We picked 16 as our voxel size because this is the best voxel size in classification performance. Too small or too large in voxel size reduces accuracy and loss in model prediction, as seen in TABLE II.

Fig. 3 displays the data acquisition procedure. Fig. 3(A) is the subject standing above the center spot. Figure 3(B) is the pole with a height of 4 meters. Figure 3(C) is the laptop. Fig.3(D) is the interface box that connects the laptop to the sensor using a LAN cable. Fig. 3 (E) is the LiDAR and Fig. 3 (F) is the cable sensor, (G) is the high beam limit, and (I) is the low beam limit.

TABLE I
OUR DATASET DISTRIBUTION FOR EACH CLASSES

Class	**Count**
Crouching	390
Hands Up	390
Lying Down	590
Sitting	390
Squatting	389
Standing	89
Total	2238

Our primary dataset contains six classes, Standing, Sitting, Lying Down, Crouching, Hands Up, and Squatting. The distribution of our dataset is exhibited in TABLE I with the sum of all data at 2238 PCD data. Since our dataset is imbalanced, we utilize over-sampling to tackle an issue in object classification.

Both ModelNet10 and ModelNet40 is a mesh data. We resampled the mesh with 2048 as the sample size to extract 3D point cloud data as in [12]. Thus, for every mesh, we have a PCD size of 2048 with its respective XYZ coordinate. Point clouds are considered unstructured data. It needs to be clarified what dimension it has. Here, we convert from a raw point cloud to a voxel grid.

B. Ablation Study

TABLE II
DIFFERENT VOXEL SIZE WITH ITS LOSS AND ACCURACY (ACC) RESULTS

Voxel Size	**Loss**	**Acc (%)**
4	0.78	74
8	**0.58**	83
12	1.07	77
16	0.64	**87**
20	1	70

Selecting the voxel size is quite challenging. A specific size of voxel affects both accuracy and loss. Over-tiny or oversized voxel leads to low accuracy and considerable loss of value. TABLE II exhibits the classification results on ModelNet10 for each voxel size. The range value of the voxel is from 4 to 20. It can be seen that the best size in terms of accuracy is 16. Eight is second to none if the performance is focused on loss reduction. However, the loss value for both 8 and 16 sizes is insignificant. According to this result, we picked 16 as our voxel size.

C. Performance Results on Object Classification

Here, we compared our model with PointNet, Effcient-Net, ResNet50, MobileNetV2, DenseNet, Xception, and VGG16Net. All of these models were assessed with overall accuracy (OA), loss, precision, recall, and F1-score.

Here, 200 epochs of training parameters are set up, and ADAM is used as the optimizer and loss function. The classification metrics' results are shown in TABLE III. With 83 %, our model leads the OA, followed by EfficientNet, Xception,

TABLE III
THE CLASSIFICATION RESULTS FROM OUR PROPOSED WORK WITH OTHER METHODS USING OUR PRIMARY DATASET. ALL METRICS EXCEPT LOSS AND NP ARE IN PERCENTAGE (%)

Model	**Loss**	**Acc**	**Prec**	**Rec**	**FS**	NP (mil)
ResNet50	29.09	18.1	8	18.3	9.7	20.88
EfficientNet	1.07	71.6	78	72	70.8	**0.12**
MobileNetV2	3.49	36.9	29	36.77	30.3	23.64
PointNet	32.08	26.41	9	17	7.8	0.75
DensetNet	6.20	24.1	34	24.3	16	7.1
Xception	25.38	44.9	60	44.3	43	20.87
VGG16	0.59	80	79.67	80	79.5	33.77
Our	**0.56**	**83**	**82.5**	**82.3**	**82.3**	0.15

MobileNetV2, and DenseNet. Our model also achieved the best value in loss, prec, rec, and FS. This performance also holds for F1-Score (FS), recall, and precision. Our loss value is also in the first quartile when standard deviation (std) and variance (var) are considered. The metrics prec, rec, and FS are computed based on the average value from each class.

D. Classification Results using Public Datasets

The training parameter setups here are 50 epochs with ADAM as an optimizer. TABLE IV displays the results of classification metrics. Our model stands at the top position in OA with 88 %, followed by PointNet, ResNet50, EfficientNet, and MobileNetV2. This performance applies too with precision (prec), recall (rec), and F1-Score (FS). Nevertheless, we failed to achieve top-notch loss compared to MobileNetV2, the difference in loss value is not entirely significant. Furthermore, our loss value is in the 1st quartile given standard deviation (std) $std = 1.338$, variance (var) $var = 1.791$. The metrics prec, rec, and FS are calculated based on their average value from each class in the dataset.

TABLE IV
CLASSIFICATION RESULTS OF THE PROPOSED MODEL WITH OTHER CLASSIFIERS USING MODELNET10 DATASET.

Model	**Loss**	**Acc**	**Prec**	**Rec**	**FS**	**NP (mil)**
ResNet50	0.57	85.7	85.5	85.1	85.2	23.64
EfficientNet	1.81	82.2	84.6	84.7	84.4	**0.12**
MobileNetV2	**0.53**	82.2	82.8	82.7	82.3	2.20
PointNet	2.57	71.9	87.3	85	85.6	0.75
DensetNet	3.60	70.5	60.8	70.7	63.8	7.00
Xception	3.79	70.3	59.2	70.5	63.4	20.88
VGG16	2.40	10	1	10	1.8	33.64
Our	0.56	**88.1**	**87.6**	**87.5**	**87.3**	0.17

EfficientNet is the least possible in the number of model parameters (NP) at 0.12, followed by our model with 0.17 and PointNet in the second and third position with 0.8, respectively. Compared with the remaining models, which have an enormous number of parameters above one million, our model is considered lightweight.

After training and evaluating using ModelNet10, we also compared all methods with ModelNet40. This dataset is a 3D-Mesh, consisting of 40 different classes, and its distribution is imbalanced. Thus, we apply the synthetic over-sampling method to this data. We tested all methods in the same environment and parameters as a fair comparison. The number of epochs is set to 50, and the learning rate (lr) is set to 0.001.

TABLE V demonstrates the result of classification performance in 8 different deep learning models, which were evaluated using six metrics, loss, acc, prec, rec, FS, and NP. Our proposed method dominated all the remaining algorithms with a loss score of 1.151, followed by MobileNetV2 and ResNet50. This fantastic performance was also attained in acc, prec, and rec with a score of 74.1 %, 75.3 %, and 72.8 %, respectively. However, for the FS, we failed to achieve the first position held by DenseNet. The most significant loss is PointNet, followed by Xception and VGG16.

TABLE V
CLASSIFICATION RESULTS OF THE PROPOSED MODEL WITH OTHER CLASSIFIERS USING MODELNET40 DATASET.

Model	**Loss**	**Acc**	**Prec**	**Rec**	**FS**	NP (mil)
ResNet50	2.29	72.3	71.3	69	68.8	23.71
EfficientNet	2.84	70.9	72.3	70.9	69.9	**0.15**
MobileNetV2	2.04	42.1	39.8	42.1	36.3	2.33
PointNet	5.98	25	7.8	11.7	7.1	0.75
DensetNet	2.57	73.8	75	72.8	**72.7**	7.13
Xception	4.24	64.5	49.8	58.5	51.9	20.95
VGG16	3.87	2.5	0.1	2.5	00.1	33.77
Our	**1.15**	**74.1**	**75.3**	**72.8**	72.3	0.37

Regarding the number of model parameters (NP), EfficientNet comes in last with 0.15, followed by our model with 0.37, and PointNet in second and third with 0.75, respectively. Our model is considered lightweight compared to the remaining models, which have a tremendous amount of parameters above one million.

V. CONCLUSION

By taking advantage of FuzzConv and DOConv, our novel approach successfully overthrew other classifiers. In our result, we set the number of voxel sizes to 16, considered an intermediate value. A smaller and larger voxel size reduces accuracy performance and increases loss score. Data imbalance significantly affects the classification task. We discovered that the class distribution for both datasets needs to be balanced. Thus, rather than reducing the data size, we apply over-sampling to achieve data balance. Our model achieved the best results for classification based on accuracy for our primary dataset, ModelNet10, and ModelNet40.

However, there are some limitations existed. The model's accuracy in object classification may be impacted due to partial coverage in LiDAR data, where objects are only partially captured in the scanned area. The evaluation of the proposed solution is limited to specific datasets like ModelNet10, ModelNet40, and a core dataset of human poses, which may raise uncertainty about its generalizability to other datasets and real-world scenarios. The choice of voxelization, transforming point clouds into 3D voxels, may introduce quantization errors and result in the loss of fine-grained information in the original point cloud data.

ACKNOWLEDGMENT

This work was supported and fully funded by the Lembaga Pengelola Dana Pendidikan (LPDP) Indonesia.

REFERENCES

[1] X. Zhao, P. Sun, Z. Xu, H. Min, and H. Yu, "Fusion of 3d lidar and camera data for object detection in autonomous vehicle applications," *IEEE Sensors Journal*, vol. 20, pp. 4901–4913, 5 2020.

[2] I. S. Weon, S. G. Lee, and J. K. Ryu, "Object recognition based interpolation with 3d lidar and vision for autonomous driving of an intelligent vehicle," *IEEE Access*, vol. 8, pp. 65599–65608, 2020.

[3] R. Soitinaho, M. Moll, and T. Oksanen, "2d lidar based object detection and tracking on a moving vehicle," vol. 55, pp. 66–71, Elsevier B.V., 2022.

[4] T. Ku, S. Galanakis, B. Boom, R. C. Veltkamp, D. Bangera, S. Gangisetty, N. Stagakis, G. Arvanitis, and K. Moustakas, "Shrec 2021: 3d point cloud change detection for street scenes," *Computers and Graphics (Pergamon)*, vol. 99, pp. 192–200, 10 2021.

[5] J. Roche, V. De-Silva, J. Hook, M. Moencks, and A. Kondoz, "A multimodal data processing system for lidar-based human activity recognition," *IEEE Transactions on Cybernetics*, vol. 52, no. 10, pp. 10027–10040, 2022.

[6] R. C. Mello, S. D. S. M., W. M. Scheidegger, M. C. Múnera, C. A. Cifuentes, M. R. Ribeiro, and A. Frizera-Neto, "The poundcloud framework for ros-based cloud robotics: Case studies on autonomous navigation and human–robot interaction," *Robotics and Autonomous Systems*, vol. 150, 4 2022.

[7] U. Stilla and Y. Xu, "Change detection of urban objects using 3d point clouds: A review," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 197, pp. 228–255, 3 2023.

[8] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3d point clouds: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, pp. 4338–4364, 12 2021.

[9] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[10] Z. Wu, S. Song, A. Khosla, Y. Fisher, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," 2015.

[11] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "Shapenet: An information-rich 3d model repository," 2015.

[12] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," pp. 77–85, 2017.

[13] M. Zaheer, S. Kottur, S. Ravanbhakhsh, B. Póczos, R. Salakhutdinov, and A. J. Smola, "Deep sets," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, (Red Hook, NY, USA), p. 3394–3404, Curran Associates Inc., 2017.

[14] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *ACM Transactions on Graphics (TOG)*, 2019.

[15] H. Zhao, L. Jiang, C.-W. Fu, and J. Jia, "Pointweb: Enhancing local neighborhood features for point cloud processing," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5560–5568, 2019.

[16] M. Joseph-Rivlin, A. Zvirin, and R. Kimmel, "Momenet: Flavor the moments in learning to classify shapes," pp. 4085–4094, 2019.

[17] R. Huang, D. Hong, Y. Xu, W. Yao, and U. Stilla, "Multi-scale local context embedding for lidar point cloud classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 17, no. 4, pp. 721–725, 2020.

[18] C. Liu, D. Zeng, A. Akbar, H. Wu, S. Jia, Z. Xu, and H. Yue, "Context-aware network for semantic segmentation toward large-scale point clouds in urban environments," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–15, 2022.

[19] Y. Liu, B. Fan, S. Xiang, and C. Pan, "Relation-shape convolutional neural network for point cloud analysis," pp. 8887–8896, 2019.

[20] J. F. Domenech, F. Escalona, F. Gomez-Donoso, and M. Cazorla, "A voxelized fractal descriptor for 3d object recognition," *IEEE Access*, vol. 8, pp. 161958–161968, 2020.

[21] F. Gomez-Donoso, F. Escalona, and M. Cazorla, "Vfkd: Voxelized fractal keypoint detector," pp. 1–8, 2022.

[22] D. Lu, Q. Xie, K. Gao, L. Xu, and J. Li, "3dctn: 3d convolution-transformer network for point cloud classification," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, pp. 24854–24865, 2022.

[23] M. J. Hsu, Y. H. Chien, W. Y. Wang, and C. C. Hsu, "A convolutional fuzzy neural network architecture for object classification with small training database," *International Journal of Fuzzy Systems*, vol. 22, pp. 1–10, 2 2020.

[24] J. Cao, Y. Li, M. Sun, Y. Chen, D. Lischinski, D. Cohen-Or, B. Chen, and C. Tu, "Do-conv: Depthwise over-parameterized convolutional layer," *IEEE Transactions on Image Processing*, vol. 31, pp. 3726–3736, 2022.

[25] S. Arora, N. Cohen, and E. Hazan, "On the optimization of deep networks: Implicit acceleration by overparameterization," vol. 80, pp. 244–253, PMLR, 2 2018.

[26] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," pp. 1912–1920, 2015.