Semantics-Augmented Quantization-Aware Training for Point Cloud Classification

Liming Huang, Yunchuan Qin, Ruihui Li†, Fan Wu† and Kenli Li

College of Computer Science and Electronic Engineering, Hunan University, Changsha, China

Abstract

Point cloud classification is a pivotal procedure in 3D computer vision, and its deployment in practical applications is often constrained by limited computational and memory resources. To address these issues, we introduce a Semantics-Augmented Quantization-Aware Training (SAQAT) framework designed for efficient and precise classification of point cloud data. The SAQAT framework incorporates a point importance prediction semantic module as a side output, which assists in identifying crucial points, along with a point importance evaluation algorithm (PIEA). The semantics module leverages point importance prediction to skillfully select quantization levels based on local geometric properties and semantic context. This approach reduces errors by retaining essential information. In synergy, the PIEA acts as the cornerstone, providing an additional layer of refinement to SAQAT framework. Furthermore, we integrates a loss function that mitigates classification loss, quantization error, and point importance prediction loss, thereby fostering a reliable representation of the quantized data. The SAQAT framework is designed for seamless integration with existing point cloud models, enhancing their efficiency while maintaining high levels of accuracy. Testing on benchmark datasets demonstrates that our SAQAT framework surpasses contemporary quantization methods in classification accuracy while simultaneously economizing on memory and computational resources. Given these advantages, our SAQAT framework holds enormous potential for a wide spectrum of applications within the rapidly evolving domain of 3D computer vision. Our code is released to this URL: https://github.com/h-liming/SAQAT.

CCS Concepts

• ***Computing methodologies*** → *Object recognition;*

1. Introduction

In the rapidly evolving domain of 3D computer vision [CCC*21], research has increasingly centered on point clouds classification, possessing considerable potential for progress. Point clouds, inherently high-dimensional and devoid of inherent structure, serve as repositories for spatial information, capturing the complex geometrical nuances that define tangible environmental landscapes. This data structure supports a wide spectrum of applications, from autonomous vehicles, robotics, and virtual reality to 3D reconstruction and beyond [RMB*08, LYWU18, WSM*18, GKF09, GKOM18]. Given its extensive utility, the demand for accurate and efficient classification of these point clouds has escalated, driving researchers and practitioners to overcome this formidable challenge.

Nevertheless, the journey towards optimal practical applications is fraught with obstacles. The imperative for real-time data processing via confined computational and storage capacities is juxtaposed against an incontrovertible principle: the efficacy of a deep learning model is commensurately linked to its computational complexity. Consequently, models with elevated predictive fidelity tend to be encumbered by sluggish operational velocities, augmented energy expenditures, and inflated requisites for both memory storage and model dimensions [LTL13]. This precarious balance acts as a potential Achilles' heel, jeopardizing the integrity and potency of classification algorithms, and thus yielding a stagnation in advancements within the discipline.

Quantization is the process of converting a large set of values into more compact ones and is a crucial step in processing point cloud data within the field of 3D computer vision applications. This data condensation facilitates efficient storage and computation, rendering data management more tractable. However, this indispensable process is not without its pitfalls. Careless or overly aggressive quantization can result in the loss of vital data, leading to quantization errors that can severely compromise classification outcomes, yielding less accurate and less reliable results. Concurrently, the task of handling voluminous point cloud datasets often comes with substantial memory requirements, posing an additional challenge in a field where computational efficiency is paramount [ZYX*21].

† Corresponding authors: Ruihui Li, liruihui@hnu.edu.cn; Fan Wu, wufan@hnu.edu.cn

In response to these multifaceted challenges, we introduce the innovative SAQAT framework, which optimizes efficiency while minimizing precision loss during point cloud classification. It achieves this through the integration of a semantic prediction module that forecasts the importance of points, coupled with a PIEA and a loss function.

The semantic prediction module, a linchpin of our framework, predicts point importance based on a variety of factors, including local geometric properties and semantic context. By leveraging these predictions as a guiding principle, the SAQAT framework can discerningly select quantization levels. This judicious selection process significantly reduces errors by preserving essential information. In conjunction, the PIEA, another key component of our model, refines the quantization process by generating point importance labels. Furthermore, the loss function is specifically designed to mitigate classification loss, quantization error, and point importance prediction loss.

Designed for adaptability, the SAQAT model can be seamlessly integrated with existing point cloud models such as PointNet [QSMG17], PointNet++ [QYSG17] and DGCNN [WSL*19], enhancing their efficiency and maintaining high levels of accuracy. We provides a comprehensive evaluation of the SAQAT model's performance, underscoring its superior classification accuracy compared to baseline methods. Given these compelling advantages, the SAQAT model holds considerable potential for broad application across diverse 3D computer vision tasks. In brief, our contributions are summarized as follows:

- We propose a SAQAT framework suitable for 3D point cloud classification, improving quantized network performance in various bit-widths.
- We design a point importance evaluation algorithm and loss function, enabling the semantic module to predict the importance of points based on local geometric properties and semantic context before quantization.
- Unlike previous quantization methods that quantize the network to the same bit-width, our framework decides whether to quantize a point's data based on it's importance, allowing both floating-point and integer data to coexist in quantization-aware training.
- Experimental results on both ModelNet40 [WSK*15] and ScanObjectNN [UPH*19] benchmarks demonstrate that our SAQAT framework can effectively and efficiently enhance the accuracy of various quantized models in point cloud classification tasks.

For clarity, the rest is organized into distinct sections. Section II provides a review of existing work on point cloud classification and quantization methods. Section III offers a detailed exposition of our SAQAT framework. Section IV presents a discussion of the experimental results, illuminating the performance of our SAQAT framework. Finally, Section V concludes the paper, summarizing our findings and suggesting potential avenues for future research in this fascinating area.

2. Related Work

The development and application of point cloud data in the realm of 3D computer vision have been the focus of numerous studies in recent years. This section presents a review of the current literature, with a specific focus on point cloud classification and quantization methods.

2.1. Point Cloud Classification

Classification of point cloud data is a fundamental task in 3D vision, with a broad range of applications including robotics, autonomous vehicles, and 3D reconstruction. Several models have been proposed to tackle this task.

A groundbreaking development in this area was PointNet [QSMG17] which directly processes point cloud data by leveraging a symmetric function to achieve permutation invariance. The PointNet framework, despite its simplicity, demonstrated impressive results in point cloud classification and segmentation tasks. The PointNet++ [QYSG17] algorithm introduced a hierarchical architecture designed for the meticulous extraction of local features at multiple scales. Employing a FPS (farthest point sampling) algorithm, PointNet++ contextualizes individual points by incorporating the attributes of their proximal neighbors. Building upon this, PointCNN [LBS*18] emerged as an extension to PointNet++, its seminal contribution being the X-Conv operation, a mechanism capable of weighting and substituting both input points and their associated features prior to subjecting them to convolutional processes.

In an attempt to capture local structures and patterns in the point cloud data, a dynamic graph-based convolutional neural network (DGCNN) [WSL*19] was introduced. The DGCNN employs EdgeConv modules that operate on local neighborhoods of points, addressing PointNet's limitation in capturing local features. The LDGCNN [ZHW*19] algorithm enhances the capabilities of DGCNN through the integration of hierarchical features sourced from multi-tiered dynamic graphs via residual connections [WZL*22]. This advancement obviates the necessity for a transformation network, while empirically demonstrating that Multi-Layer Perceptrons (MLPs) [Hor91] are efficacious in the extraction of transformation-invariant characteristics. PointMLP [MQY*22] employs a pure residual network architecture and integrates a lightweight geometric affine module, significantly improving inference speed. DeLA [CXZ*23] presents a method to decouple spatial relations from local aggregation, demonstrating that simple neighbor pooling can effectively fuse features, improving speed and performance.

Transformer [VSP*17] have achieved remarkable success in the 2D vision field, and recently, numerous researchers have extended this architecture to the processing of point clouds. The initial successful works in this direction were PCT [GCL*21] and Point Transformer [ZJJ*21]. PCT [GCL*21] introduced offset-attention that incorporates an implicit Laplace operator and a normalization refinement, offering permutation invariance. Point Transformer [ZJJ*21] utilized local vector attention to save on computational resources, overall resembling a deeper PointNet++ with favorable results. Point-MAE [PWT*22], Point-BERT [YTR*22] and Point-GPT [CWY*24] focus on using pre-trained models to perform

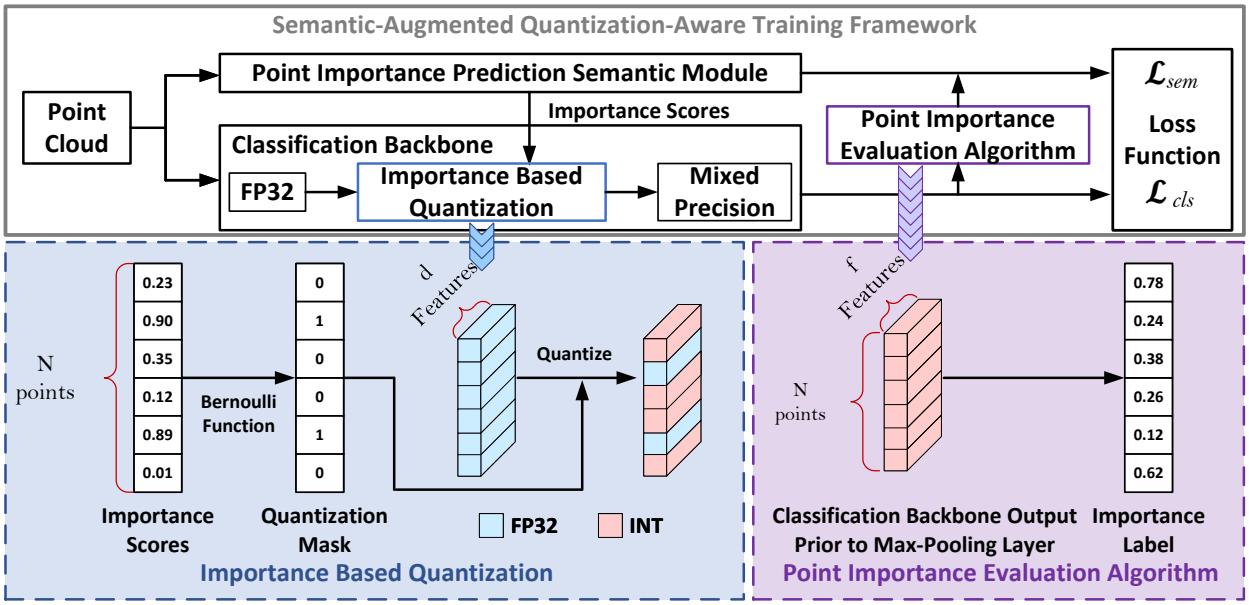

Figure 1: *Illustration of the proposed SAQAT framework. For an inputted point cloud, it is first fed into the point importance prediction semantic module, from which the importance scores for each point are obtained. When the point cloud is input into the classification backbone, the quantization mask generated based on the importance scores is used to select the quantization precision. The PIEA algorithm is used before the max-pooling layer of the classification backbone to obtain the importance label.*

masked modeling of point clouds to learn more effective feature representations.

However, these methods, while having contributed significantly to the progress in point cloud classification, do not directly address issues related to quantization, which can have a substantial impact on the model's computational efficiency and accuracy.

2.2. Quantization for Deep Neural Networks

In deep neural networks, the exploration and application of quantization techniques have been extensive. These methods are crucial in reducing the model's size and enhancing the speed of inference, without altering the model's original structure. An extensive corpus of scholarly investigations has been undertaken to elucidate the effect of quantization across various bit-widths [CBD15, HMD15, LUW17] and data representation formats [MNA*17, CLK*19, KMM*19]. Quantization-aware training (QAT) has risen to prominence as a standard method for crafting sturdy quantized models with a minimal margin of error [WLL*19, ZBIW19]. Core QAT methodologies simulate the numerical imprecisions that arise as collateral effects of the quantization process during the forward pass [JKC*18]. These techniques employ the Straight-Through Estimator (STE) [BLC13] to calculate gradients, operating under the premise that the quantization effects are inconsequential. Degree-Quant [TFML20] developed a tailored quantization approach for graph neural network (GNN), which involves creating masks for high-degree nodes and quantizing only those nodes that are unmasked, effectively mitigating the issue of substantial degree variation between nodes. The precision loss of GNNs is almost negligible when quantized to 8-bit using Degree-Quant [TFML20]. However, when quantized to lower bit, the accuracy of this method significantly decreases.

Several quantization methods with prediction-head structure and using feature maps as quantization metrics have been introduced for 2D computer vision tasks. SeerNet [CMX*19] introduces a novel method to speed up CNN inference by leveraging feature map sparsity, using a highly quantized version of the original network to accurately predict output sparsity. However, this work merely accelerates the inference speed of CNNs that have already been quantized and does not involve QAT. DynamicQuant [LWH*22] presents a novel approach to deep neural network quantization, where the quantization bit-width of each layer dynamically adjusts based on the specific image being processed. Although this technique acknowledges the diversity and complexity of natural images and allows for customized quantization, it is not applicable to 3D computer vision tasks due to the inherent differences in data structure between 2D images and 3D point clouds.

Recently, a number of advancements have been made in QAT techniques, particularly focusing on enhancements to the STE. APQ [YLS*21] presents a new training approach for neural networks that allows for dynamic adjustment of numerical precision during inference, facilitating a real-time trade-off between computational efficiency and accuracy across various vision tasks without performance degradation. EWGS [LKH21] introduces Element-wise Gradient Scaling (EWGS), an alternative to the STE, for training quantized neural networks by adaptively scaling each gradient element to account for discretization errors, thereby improving both stability and accuracy. LLT [WDW*22] introduces learnable lookup tables for neural network quantization, offering an end-to-

end trainable and computationally efficient approach that outperforms traditional linear quantizers across multiple tasks.

To address the resource limitations of point cloud applications on edge devices, several quantization techniques for point clouds have been developed. BiPointNet [QCZ*20] introduces the pioneering binarization approach for optimizing deep learning with point clouds, employing Entropy-Maximizing Aggregation and Layer-wise Scale Recovery to significantly surpass contemporary binarization methods. LiDAR-PTQ [ZLZ*24] is a tailored Post-Training Quantization method that identifies the underlying reasons behind performance decline in quantizing LiDAR-based detection, offering favorable efficiency compared to conventional QAT approaches.

2.3. Point Cloud Saliency Map

In the field of 2D computer vision, saliency map has been widely studied to assess which pixels are more important for model classification performance [SVZ13, PMJ*16]. Saliency map has also been proposed as a crucial technique in point clouds learning to mitigate noise and reduce data dimensions resulting from their unstructured nature, enhancing both model robustness and computational efficiency [ALM19, ALM20]. In [ZCY*19], they sequentially relocate points to the center of the point cloud and construct a saliency map by determining their importance for classification based on their impact on the classification loss. In [SYX*21], they developed a technique employing a projection neural network to generate a saliency map. SD-GCN [ML22] creates a saliency map by measuring the distance between the normal of each point's features and the principal normal of the input point cloud.

The existing point cloud saliency methods share certain similarities with our PIEA, but also differ notably. The similarity is that both can only calculate the point importance after the point cloud backbone network has produced predictions. The difference lies in that these methods rely on pre-trained point cloud networks and gradients, whereas our algorithm operates independently of these requirements.

3. Methodology

Fig. 1 provides a comprehensive depiction of the SAQAT framework proposed in this study. In Section 3.1, our discourse initially unravels the SAQAT framework. We then discuss three key features of our framework that make it particularly effective. They are: the point importance prediction semantic module, the PIEA, and the loss function. These are described in detail in Sections 3.2, 3.3, and 3.4, respectively, illuminating their individual and collective roles in advancing the state of point cloud classification.

3.1. SAQAT Framework for Point Cloud Classification

Consider a point cloud P, containing N points and represented as $P \in \mathbb{R}^{N\times 3}$. Inspired by [TFML20], we propose a novel SAQAT mechanism to quantize the point cloud. Notably, our work stands apart from typical QAT architectures as it integrates an auxiliary semantic branch that enables real-time prediction of the importance of points within the point cloud. With this foresight, we can assign

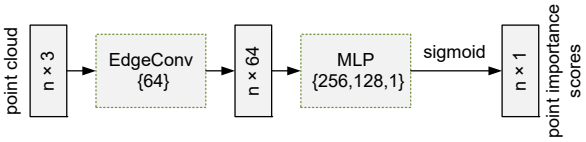

Figure 2: *Block diagram of the point importance prediction semantic module.*

full precision to semantically important points, thus preserving critical data while compressing relatively unimportant points to reduce the overall bit-width.

Our network architecture bifurcates into two main branches-the classification backbone for 3D classification tasks, and a more streamlined network for predictive functions. During training, the point cloud P serves as an input to both the classification backbone and the semantic prediction branch. We first route each point cloud through the point importance prediction semantic module (elucidated in Section 3.2). Consequently, the semantic branch yields a continuous $[0,1]$ vector of importance, $\mathbf{p}$, equal in length to the number of points for each point cloud. This vector indicates the degree of semantic information each point holds. According to Theorem 2 proposed in the PointNet [QSMG17] paper, after the point cloud passes through the maxpooling layer of the backbone network, only the features of the points in the *critical point set* are retained. We consider the points with more retained features to be more important, contributing more significantly to the final classification result. In our model, a higher value of p_i corresponds to greater importance of point i, given that points of high importance are typically the primary contributors to classification backbone errors. It's important to note that such semantic prediction is performed only once at the input position of each point cloud frame.

Regarding the classification backbone, prior to performing quantization, we use a hyperparameter p_{max} which, when multiplied with the point importance vector $\mathbf{p}$, produces a vector of masking probabilities $\mathbf{q}$. Subsequently, this probability vector, $\mathbf{q}$, is converted into a protective point mask $\mathbf{m}$, a $\{0,1\}$ vector, using a Bernoulli function: $\mathbf{m} \sim \text{Bernoulli}(\mathbf{q})$. Owing to the sparse nature of point clouds and the operational characteristics of 3D sparse convolution, we observe less diffusion of semantic information in 3D point clouds compared to 2D images. Therefore, we pass this point mask from the first layer to the last layer of our network, keeping the prediction overhead minimal. SAQAT is designed to promote precise transfer of semantic information by probabilistically protecting important points from quantization. All masked points maintain full precision during processing in the classification backbone. During testing, protection is deactivated, allowing all points to operate at low precision. We adopt this training strategy because the quantization errors of important points can produce large backpropagation gradients, leading to unstable updates of the model weights. The use of a Bernoulli function during training ensures that most important points maintain FP32 precision in each epoch, with only a small fraction being quantized. This allows for smoother updates to the model weights. Over 200 epochs, almost

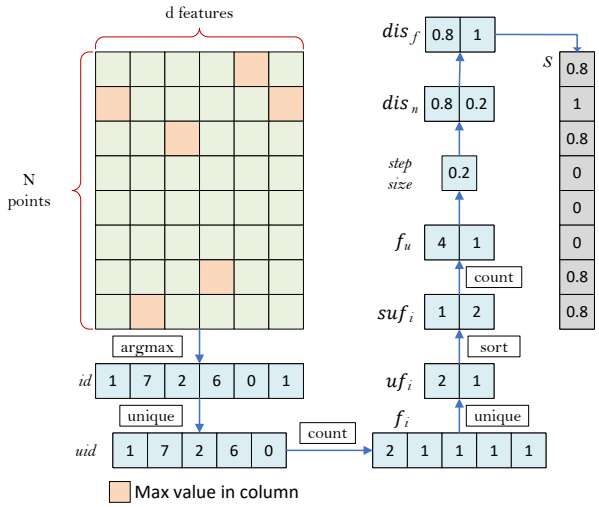

Figure 3: *llustration of the proposed PIEA.*

every important point is quantized several times, resulting in better outcomes on the test set when fully quantized.

In this study, we emphasize point cloud based 3D object classification, a crucial aspect for 3D computer vision applications. Accordingly, we adopt the conventional classification loss, $\mathcal{L}_{cls}$, for our QAT framework. However, to further enhance the quality of semantic prediction, we incorporate a PIEA (elaborated in Section 3.3), and an additional semantic prediction loss $\mathcal{L}_{sem}$. For the i-th point, we assign an importance label $s_i \in [0,1]$ using the PIEA, and subsequently train the prediction p_i to approximate s_i.

3.2. Point Importance Prediction Semantic Module

Since PIEA can only obtain the point importance label after the point cloud has been processed by the classification backbone, and we wish to obtain the point importance score at the time of point cloud input so that the point importance score can be utilized in the quantization of the classification backbone, therefore, we designed the point importance prediction semantic module.

We initiate the procedure by assimilating local geometric traits and semantic context corresponding to each point within the point cloud. This process is enabled through the utilization of a variation of EdgeConv, which serves as a foundational filtration CNN to transform the rudimentary input of the point cloud into a feature-rich output. The block diagram of point importance prediction semantic module is shown in Fig. 2. Contrasting starkly with the standard EdgeConv operator that harnesses a duo of kernels in the edge feature function, our approach deploys an triple-kernel formula: $h_\Theta(x_i, x_j - x_i, (x_j - x_i)^2)$ following [NTR*20]. This formula encompasses an element-wise square operation between each individual point, denoted as x_i, and its neighbouring point, represented as x_j.

Within the point importance prediction semantic module, applying EdgeConv equipped with 64 filters to the input point cloud of dimensions $n \times 3$ yields a transformed point cloud featuring dimensions of $n \times 64$. Following this, a chain of MLP layers, size 256, 128 and 1 respectively, transform the feature vector into the importance score. The concluding stage involves the application of a sigmoid function, which regulates the importance score to fall within the confines of 0 and 1. Additionally, we apply the LeakyReLU activation function and batch normalization across all layers to enhance the module's performance.

Algorithm 1 Point Importance Evaluation Algorithm (PIEA)

Input: $\mathbf{F} \in \mathbb{R}^{N \times d}$
Output: $\mathbf{S} \in \mathbb{R}^{N}$
for $i = 0$ **to** ncols($\mathbf{F}$) $- 1$ **do**
  $\mathbf{id}[i] = \text{argmax}(\mathbf{F}[:, i])$
end for
$\mathbf{uid} = \text{unique}(\mathbf{id})$
$\mathbf{f_i}[j] = |\{i \mid \mathbf{id}[i] == \mathbf{uid}[j]\}|$ // frequency of $\mathbf{uid}[j]$ in $\mathbf{id}$
$\mathbf{uf_i} = \text{unique}(\mathbf{f_i})$
$\mathbf{suf_i} = \text{sort}(\mathbf{uf_i})$
$\mathbf{f_u}[k] = |\{j \mid \mathbf{f_i}[j] == \mathbf{suf_i}[k]\}|$ // frequency of $\mathbf{suf_i}[k]$ in $\mathbf{f_i}$
step_size $= 1 \div \text{sum}(\mathbf{f_u})$
$\mathbf{dis_n} = \text{step\_size} \times \mathbf{f_u}$
$\mathbf{dis_f} = \text{cumsum}(\mathbf{dis_n})$
for $i = 0$ **to** $|\mathbf{uid}| - 1$ **do**
  $\mathbf{S}[\mathbf{uid}[i]] = \mathbf{dis_f}[\mathbf{suf_i}.\text{index}(\mathbf{f_i}[i])]$
end for

3.3. Point Importance Evaluation Algorithm

The training of the point importance prediction semantic module necessitates the existence of the importance label s_i. [QSMG17] provides the definition of the critical point set. Identifying these key points within a point cloud, which embody the maximal data that ought to be safeguarded within a max pooling procedure, forms the crux of this process. Such points may be susceptible to variations depending upon the task or application at hand. The data fed into the max pooling layer comprises an unordered point cloud embodying N points, wherein each is expressed through a feature vector $x \in \mathbb{R}^d$, with $\mathbb{R}$ denoting the set of real numbers and d representing the dimension of the feature vector. The PIEA endeavors to allocate importance label $s_i \in [0,1]$ to every input point.

A visualization of the proposed PIEA can be found in Fig. 3, with the pseudo-code for PIEA elaborated upon in Algorithm 1. The following elucidates the algorithm steps:

1. The input point cloud $\mathbf{F}$ is essentially a matrix comprising n rows (equating to n input points) and d columns (paralleling to d-dimensional feature vectors).
2. The first operation (Operation 2) entails the storage of the index of each row featuring maximum value into the index vector **id**, which houses the indices of all points contributing to the feature vector. These points, by definition, are labeled as important points.
3. **id** might encapsulate multiple iterations of the same point. To mitigate these repetitions, unique indices are extracted from **id** through the "unique" function (Operation 4). The output set of unique indices is denoted as the important set **uid**. The frequency

of each important point's occurrence in **id** is designated as $\mathbf{f_i}$ (Operation 5).

4. Operation 6 involves the extraction of unique frequency from $\mathbf{f_i}$, expressed as $\mathbf{uf_i}$. Following this, $\mathbf{uf_i}$ is arranged in ascending order to yield $\mathbf{suf_i}$ (Operation 7). The frequency of each unique frequency's presence within $\mathbf{f_i}$ is accounted for (Operation 8), and is represented as $\mathbf{f_u}$.
5. Subsequent operations involve the calculation of the importance score step size, equivalent to the reciprocal of sum($\mathbf{f_u}$) (Operation 9). Number-based importance distributions $\mathbf{dis_n}$ are derived by multiplying $\mathbf{f_u}$ by the step size (Operation 10).
6. Frequency-based importance distributions, denoted as $\mathbf{dis_f}$, are obtained by executing cumsum($\mathbf{dis_n}$) (Operation 11).
7. Finally, the frequency of each point in **uid** is identified, with the index of the frequency in $\mathbf{suf_i}$ located. The importance score **S** is then directly assigned using $\mathbf{dis_f}$ and the index (Operation 13). It should be noted that all points deemed unimportant are allocated **0** as their importance score.

3.4. Defining the Loss Function

Training the point importance prediction semantic module solely utilizing the basic loss functions related to the classification task proves inadequate. Consequently, we formulate a loss function, $\mathcal{L}$, as follows:

$$\mathcal{L} = \mathcal{L}_{cls} + \alpha\mathcal{L}_{sem}, \tag{1}$$

where $\mathcal{L}_{cls}$ symbolizes the classification loss, while $\mathcal{L}_{sem}$ signifies the loss of the point importance prediction semantic module. The α term serves as a weighting factor to equilibrate the impacts of both terms. To elaborate, $\mathcal{L}_{cls}$ corresponds to the original backbone network classification loss. We employ a binary cross-entropy loss computation for $\mathcal{L}_{sem}$, which gauges the disparity between the predicted importance score and the importance label. As for α, it constitutes the hyperparameter for fine-tuning each individual backbone network and dataset.

4. Experiments

In this section, we evaluate our SAQAT framework by comparing it with standard datasets and current leading quantization methods, including APQ [YLS*21], EWGS [LKH21], and LLT [WDW*22]. To explicitly describe the quantization of weights and activations as INT8 or INT4, etc., we use labels such as "8-bit" or "4-bit". Initially, we describe the experimental arrangement, which will be succeeded by a comprehensive analysis of the derived outcomes.

4.1. Experimental Setup

Datasets: Our assessment is performed using the ModelNet40 [WSK*15] and ScanObjectNN [UPH*19] 3D object classification dataset. The ModelNet40 dataset encompasses 12,311 CAD models that have been meshed, hailing from 40 distinct object categories. Among these, 9,843 models are employed for training purposes, while the remaining 2,468 are dedicated to testing. The ScanObjectNN dataset comprises approximately 15,000 real-world scanned objects, organized into 15 distinct categories featuring 2,902 unique instances of objects. We focus on the most challenging variant, known as PB_T50_RS. This variant is particularly demanding due to the presence of occlusions and noise, presenting considerable obstacles for current methodologies in point cloud classification. For the two datasets, points numbering 1,024 are uniformly extracted from each model's mesh surface and then normalized to fit within the confines of a unit sphere.

Implementation Specifics: Our framework employs PointNet [QSMG17], PointNet++ [QYSG17] and DGCNN [WSL*19] as backbone networks, alongside STE [CHS*16] for quantization methods. PointNet [QSMG17] and PointNet++ [QYSG17] serves as the underlying architecture for a multitude of point-based 3D methodologies [YSLJ20, LZC*21], while DGCNN [WSL*19] finds its applications in an array of graph-centric 3D techniques [LNL*19, XSW*20]. We have opted for PyTorch renditions of PointNet [Yan19], PointNet++ [QYSG17] and DGCNN [Tao20], and have harnessed the prowess of Brevitas [Pap23] as a library to conduct quantization-aware training for neural network quantization.

Each layer of the backbone network is subjected to quantization protocols, encompassing both the first and last layers. When 8-bit quantization is employed, both the input, weights and activations of the model are quantized into 8-bit integers. However, under the regime of 4-bit, 3-bit and 2-bit quantization, the input, weights and activations of the model's first layer are quantized into 8-bit integers, while the remaining components are quantized to 4-bit , 3-bit and 2-bit respectively. Our approach, along with the three comparative methods, all adopt this strategy for a fair comparison. This stratagem was devised upon the realization that it strikes an efficacious balance, augmenting the model's accuracy without incurring a substantial computational overhead [CLT*20]. We employ the AbsMax method to monitor the quantization range, a technique that is both expeditious and minimizes the introduction of substantial quantization errors. For all output from activation functions, we implement Per-Tensor quantization: a uniform quantization operation is executed across the entire tensor, with all elements sharing a single scaling factor. Conversely, for all weights in convolutional and linear layers, we adopt Per-Channel quantization: this signifies that each channel—or equivalently, each input feature—possesses an individual scaling factor. This form of quantization is markedly more precise as it allows for channel-specific adjustments to the quantization parameters in order to maximize accuracy. Nevertheless, this entails the necessity for storing a larger array of quantization parameters, which could potentially augment the overall size of the model.

The models are trained from scratch, eschewing the utilization of pre-trained full-precision models. This training configuration follows [ZST*18, FSG*20], which can assist the network in dealing with quantization noise when the network is partially quantized. This setup minimizes the quantization errors associated with the use of the STE, resulting in better outcomes compared to fine-tune a pre-trained model, particularly when quantizing to lower than 8-bit. We utilize the Adam optimizer for the training of PointNet and PointNet++, with the learning rate and decay rate set at 5e-4 and 1e-4 respectively. PointNet training is conducted on an Nvidia RTX2080Ti GPU with a batch size of 32, requiring approximately 7 hours to complete 200 epochs. PointNet++ training is performed

Table 1: *Classification performance achieved on the ModelNet40 and ScanObjectNN datasets in overall accuracy (OA, %).*

Model	Method	ModelNet40					ScanObjectNN				
		FP32	8-bit	4-bit	3-bit	2-bit	FP32	8-bit	4-bit	3-bit	2-bit
PointNet	APQ [YLS*21]	90.6	90.6	**90.6**	89.8	**89.3**	68.2	66.7	66.0	65.8	**65.4**
	EWGS [LKH21]		85.9	85.6	75.2	50.9		59.3	49.1	38.7	24.9
	LLT [WDW*22]		78.7	74.2	69.3	65.4		48.5	50.3	46.6	43.6
	SAQAT(Ours)		**90.7**	**90.6**	**90.3**	88.8		**68.0**	**68.4**	**66.3**	64.9
PointNet++	APQ [YLS*21]	92.2	92.0	92.0	91.9	91.4	79.8	77.3	77.6	78.3	76.7
	EWGS [LKH21]		91.2	90.7	90.3	34.3		76.8	74.2	70.5	20.5
	LLT [WDW*22]		90.0	89.2	89.1	81.0		70.5	68.8	68.3	62.6
	SAQAT(Ours)		**92.5**	**92.6**	**92.3**	**91.7**		**79.4**	**79.5**	**78.4**	**78.0**
DGCNN	APQ [YLS*21]	93.3	88.5	82.5	81.8	75.9	78.1	72.8	74.6	70.2	59.5
	EWGS [LKH21]		86.1	77.1	41.3	25.8		42.2	32.5	33.6	20.2
	LLT [WDW*22]		91.2	89.7	78.8	74.9		68.8	66.9	65.7	58.7
	SAQAT(Ours)		**92.8**	**92.5**	**91.9**	**89.3**		**79.9**	**78.0**	**76.2**	**67.6**

Table 2: *8-bit latency results run on a 22 core 2.1GHz Intel Xeon Gold 6152 CPU. Quantization provides large speedups.*

Model	Model Size (KB)			Inference Time (ms)		
	FP32	8-bit	Ratio	FP32	8-bit	Ratio
PointNet	2721	718	3.79	27	8	3.38
PointNet++	5751	1459	3.94	66	14	4.71
DGCNN	5036	1314	3.83	2812	705	3.99

on an Nvidia RTX4090 GPU, with a batch size of 32, necessitating 10 hours to finalize 200 epochs. For DGCNN, the SGD optimizer is used with a learning rate and decay rate of 1e-3 and 1e-5 respectively. The network training, conducted on an Nvidia RTX3090 GPU with a batch size of 32, requires 15 hours for completion of 250 epochs. To effectuate an equitable comparative analysis, we have conducted a modest hyperparameter search across all SOTA methodologies, inclusive of learning rates and their idiosyncratic quantization parameters.

4.2. Results and Discussion

4.2.1. Evaluation of Classification Accuracy

Table 1 illustrates the superior quantization capabilities of SAQAT across three backbone models, two datasets, and multiple quantization bit-widths. Nearly all results obtained from our SAQAT surpass other SOTA methods, as shown in the last row of each model. The best results are highlighted in bold. We observe that the APQ [YLS*21] performs well on PointNet and PointNet++ but poorly on DGCNN. Conversely, the LLT [WDW*22] exhibits the opposite behavior. The EWGS [LKH21] exhibits moderate performance across three backbone networks, but its performance plummets dramatically when the networks are quantized to 2-bit. However, our SAQAT quantization method demonstrates the highest applicability, achieving the best results in almost all tests. In detail, for PointNet, our method exhibits slightly lower classification accuracy than APQ by 0.5% only in 2-bit quantization, while accuracy at other bit-widths surpasses that of other methods, with a maximum lead of 2.4% over the second best approach. For PointNet++, it shows the minimal reduction in performance across any bit-width, our method yields results that are slightly ahead of the second best approach. For the test results of DGCNN, our method significantly outperforms other quantization methods under various bit-widths. Compared to the second-best method (*i.e.*, APQ), for the ModelNet40 dataset, our method achieves 11.1% and 13.4% higher accuracy when quantized to 3-bit and 2-bit, respectively. On the ScanObjectNN dataset, when quantized to 8-bit, 3-bit, and 2-bit, our method exceeds its accuracy by 7.1%, 6%, and 8.1%, respectively.

In our framework configuration, we adopted an aggressive approach: quantization is applied across all network layers. When quantizing the first layer of the network, the features of the input points are solely their three-dimensional coordinates. Quantization leads to the homogenization of coordinates for some adjacent points. However, our framework preserves the salient information of pivotal points during the training process, thereby augmenting the overall accuracy of the entire network. To achieve optimal inference acceleration, it is imperative to quantize each layer. This ensures that the model consistently operates using integer-based arithmetic, thereby eliminating the need for floating-point computations.

4.2.2. Evaluation of Classification Efficiency

In addition to significantly reducing model size, quantization also serves to expedite model inference time and enhance throughput, particularly when employing CPUs for inference [XLC*19]. Our observations substantiate that the 8-bit algorithm can amplify inference speed by up to 4.71x. It's noteworthy that the multiplicative increase in inference speed is contingent upon the depth of the backbone network. Ergo, we conducted benchmark tests on PointNet, PointNet++ and DGCNN, with the results tabulated in Table 2. For PointNet, the model size was contracted by 3.79x, yielding an acceleration of 3.38x. For PointNet++, the model size was re-

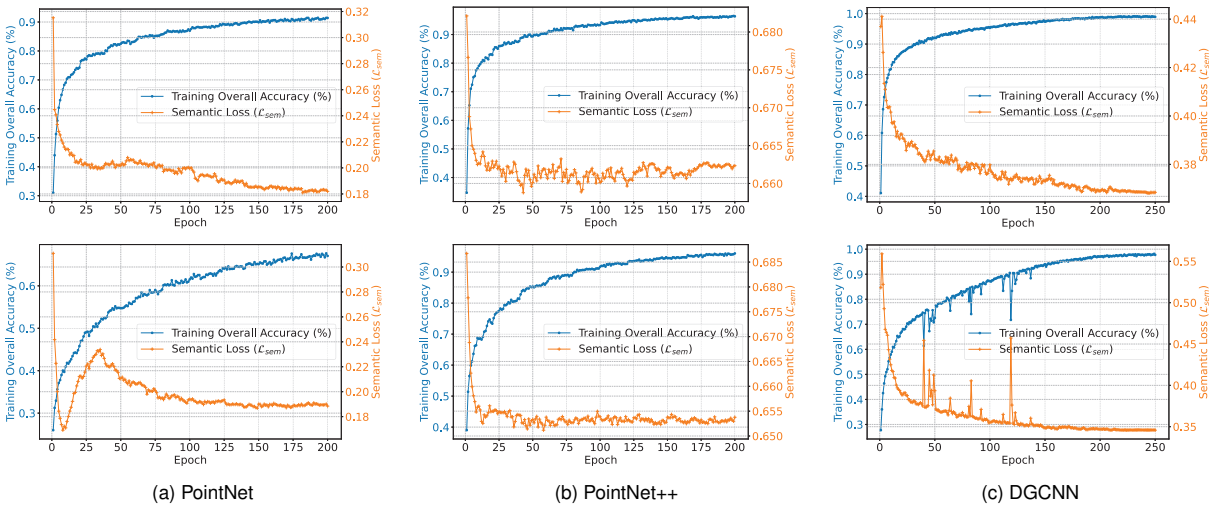

Figure 4: *Effectiveness test of the point importance prediction semantic module. The metric is training overall classification accuracy (%) and semantic loss ($\mathcal{L}_{sem}$) over epochs under 8-bit quantization. The dataset in the first row is ModelNet40, while that in the second row is ScanObjectNN. The models in each column are: (a) PointNet. (b) PointNet++. (c) DGCNN.*

Table 3: *Effectiveness test of the PIEA. Experimental results on the overall accuracy (%) of different point importance evaluation methods under 8-bit quantization.*

Model	Method	ModelNet40	ScanObjectNN
PointNet	F_{mean}	90.1	65.5
	$PIEA_{bin}$	90.4	66.6
	PIEA (Ours)	**90.7**	**68.0**
PointNet++	F_{mean}	92.0	78.8
	$PIEA_{bin}$	92.3	78.5
	PIEA (Ours)	**92.5**	**79.4**
DGCNN	F_{mean}	92.3	78.9
	$PIEA_{bin}$	92.7	79.8
	PIEA (Ours)	**92.8**	**79.9**

duced by 3.94x, resulting in an acceleration of 4.71x. For DGCNN, the model was shrunk by 3.83x, engendering an acceleration of 3.99x. Equally salient is the imperative of deploying quantized networks for optimal utility on compact devices such as smartphones. These accelerators predominantly expedite integer calculations and CPUs remain the ubiquitous choice for model servicing on servers [LLLB17]. The diminution of CPU latency is attributable to enhanced cache performance of sparse operations; in contrast, GPUs offer lesser quantization advantages due to their massive parallel architectures adopting non-cache strategies to mask the intrinsic sluggish random memory accesses in such applications.

4.3. Ablation Study

In this section, a series of ablation studies were conducted to examine the contributions of the proposed point importance prediction semantic module, PIEA, and the introduced loss function in enhancing the performance of classification tasks for quantized networks. Due to the considerable computational cost and extended training time, we opted to utilize 8-bit quantization for these ablation studies.

4.3.1. Effectiveness of Point Importance Prediction Semantic Module

Our SAQAT framework harnesses point importance prediction to adeptly select quantization levels, contingent on local geometric features and semantic context. As a result, the prediction accuracy becomes pivotal for the succeeding quantization-aware training. We monitor the predictive loss of the semantics module ($\mathcal{L}_{sem}$) throughout the training process, with the outcomes displayed in Fig. 4. In this experiment, a notable trend is observed with the increase in training epochs: the $\mathcal{L}_{sem}$ gradually decreases, while the overall training accuracy steadily improves. However, the decreasing trend of $\mathcal{L}_{sem}$ for the three backbone networks shows slight variations; PointNet and DGCNN exhibit a relatively steady decline in $\mathcal{L}_{sem}$ on the ModelNet40 dataset, while other sets of experiments show some oscillations. The reason is that the ScanObjectNN dataset makes it more difficult for the semantic module to capture crucial point information, and in PointNet++, the SA (Set Abstraction) module ultimately retains only 128 points for training the semantic module, whereas the other two backbone networks utilize all 1024 points in each iteration. The experimental results demonstrate that the accuracy of importance predictions made by our semantic module continuously strengthens throughout the learning process, thereby enhancing the backbone model's ability to classify samples. Consequently, this leads to an improvement in the quantized model's accuracy.

4.3.2. Effectiveness of Point Importance Evaluation Algorithm

The function of the PIEA lies in the computational generation of a scalar metric of point importance, symbolically represented by

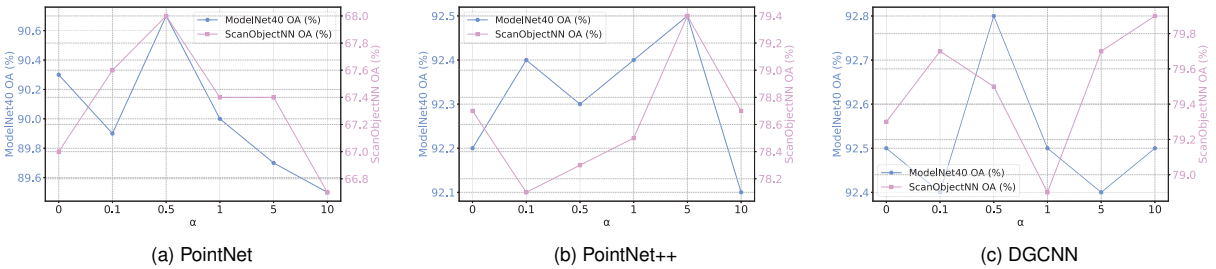

Figure 5: *Effectiveness test of the loss function. The metric is overall classification accuracy of 8-bit quantization on the ModelNet40 and ScanObjectNN test sets. The left Y-axis represents accuracy on the ModelNet40 dataset, and the right Y-axis represents accuracy on the ScanObjectNN dataset. The three backbone networks have different optimal* α *values.*

s_i, which serves as an indispensable catalyst in the training process of the semantic module. To substantiate the efficacy of the PIEA approach, we juxtapose it against two rudimentary alternatives: F_{mean} and $PIEA_{bin}$. F_{mean} simply adds up the feature vectors of each point and then divides by the dimensionality of the feature vectors to calculate the average value of the features. Subsequently, based on the size of the average value, it uniformly maps the importance of each point to a range from 0 to 1 using Min-Max normalization. However, this strategy has a drawback: since backbone networks utilize max pooling as a symmetric function to extract features, it results in assigning greater importance to points with larger average values but no prominent features in any dimension. Conversely, points with a few standout feature dimensions, which should have a medium level of importance, are assigned minimal importance by this strategy. This leads to a discrepancy between the assigned importance and the actual expected importance. $PIEA_{bin}$ employs a dichotomous assignment for s_i. Under this alternative schema, a specific point within the point cloud acquires the label “important" if any one of its attributes influences the maximal pooling operational output—this is irrespective of the cumulative cardinality of such contributive features. To elucidate, assume one point contributes a pair of features, while another contributes a decuple; in the context of $PIEA_{bin}$, both instances are accorded equivalent importance. Simplifying, within the confines of $PIEA_{bin}$, the construct of “importance" is binarized; a point is rendered as either important or not. Table 3 illustrates the comparative outcomes, confirming that under 8-bit quantization, in the ModelNet40 dataset, PIEA enhanced the prediction accuracy of PointNet by 0.3%, PointNet++ by 0.2%, and DGCNN by 0.1%. In the ScanObjectNN dataset, PIEA increased the prediction accuracy of PointNet by 1.4%, PointNet++ by 0.6%, and DGCNN by 0.1%. This thereby corroborates its role in enhancing the overall accuracy of the quantized backbone network.

4.3.3. Effectiveness of Loss Function

To validate the effectiveness of the loss function, we conducted experiments on three types of backbone networks using multiple α values under 8-bit quantization, and Fig. 5 displays the results. When α isn't zero, the value of the semantic module's prediction error factors into the calculation of the loss function. The larger the α value, the more our framework emphasizes the prediction of the semantic module. The smaller the α value, the more our framework prioritizes the classification accuracy of the backbone networks. When the α value is 0, backpropagation only uses the classification loss of the backbone networks. The experimental results indicate that for PointNet and PointNet++, the optimal α values are 0.5 and 5, respectively, on both datasets. For DGCNN, an α value of 0.5 yields the best results on the ModelNet40 dataset, and an α value of 10 is most effective on the ScanObjectNN dataset. The comparison data indicate that our loss function is particularly suitable for the methodology described in this paper.

5. Conclusion

In conclusion, this work presented SAQAT, a novel framework for efficient quantization of point cloud classification networks. We adopted a unique approach incorporating a point importance prediction semantic module and a point importance evaluation algorithm, leading to enhanced effectiveness. Our loss function, which combines backbone network classification loss, quantization error and semantic module prediction loss, proved essential in fine-tuning the model's performance. Evaluations on benchmark datasets highlighted the superiority of SAQAT over SOTA methods, with significant accuracy improvements. While we demonstrated the potency of SAQAT in this study, future research could explore its applications across a broader range of tasks and architectures. Ultimately, the findings from this work contribute to the advancement of 3D computer vision, an area of crucial importance in fields ranging from robotics to autonomous driving.

6. Acknowledgments

The work is supported by the Science and Technology Innovation 2030 - “New Generation Artificial Intelligence” Major Project (2021ZD40300), National Natural Science Foundation of China (No.62202151).

References

[ALM19] ARVANITIS G., LALOS A. S., MOUSTAKAS K.: Saliency mapping for processing 3d meshes in industrial modeling applications. In *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)* (2019), vol. 1, IEEE, pp. 683–686. 4

[ALM20] ARVANITIS G., LALOS A. S., MOUSTAKAS K.: Robust and fast 3-d saliency mapping for industrial modeling applications. *IEEE Transactions on Industrial Informatics 17*, 2 (2020), 1307–1317. 4

[BLC13] BENGIO Y., LÉONARD N., COURVILLE A.: Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432* (2013). 3

[CBD15] COURBARIAUX M., BENGIO Y., DAVID J.-P.: Binaryconnect: Training deep neural networks with binary weights during propagations. *Advances in neural information processing systems 28* (2015). 3

[CCC*21] CUI Y., CHEN R., CHU W., CHEN L., TIAN D., LI Y., CAO D.: Deep learning for image and point cloud fusion in autonomous driving: A review. *IEEE Transactions on Intelligent Transportation Systems 23*, 2 (2021), 722–739. 1

[CHS*16] COURBARIAUX M., HUBARA I., SOUDRY D., EL-YANIV R., BENGIO Y.: Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830* (2016). 6

[CLK*19] CARMICHAEL Z., LANGROUDI H. F., KHAZANOV C., LILLIE J., GUSTAFSON J. L., KUDITHIPUDI D.: Deep positron: A deep neural network using the posit number system. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)* (2019), IEEE, pp. 1421–1426. 3

[CLT*20] CHEN C., LI K., TEO S. G., ZOU X., LI K., ZENG Z.: Citywide traffic flow prediction based on multiple gated spatio-temporal convolutional neural networks. *ACM Transactions on Knowledge Discovery from Data (TKDD) 14*, 4 (2020), 1–23. 6

[CMX*19] CAO S., MA L., XIAO W., ZHANG C., LIU Y., ZHANG L., NIE L., YANG Z.: Seernet: Predicting convolutional neural network feature-map sparsity through low-bit quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 11216–11225. 3

[CWY*24] CHEN G., WANG M., YANG Y., YU K., YUAN L., YUE Y.: Pointgpt: Auto-regressively generative pre-training from point clouds. *Advances in Neural Information Processing Systems 36* (2024). 2

[CXZ*23] CHEN B., XIA Y., ZANG Y., WANG C., LI J.: Decoupled local aggregation for point cloud learning. *arXiv preprint arXiv:2308.16532* (2023). 2

[FSG*20] FAN A., STOCK P., GRAHAM B., GRAVE E., GRIBONVAL R., JEGOU H., JOULIN A.: Training with quantization noise for extreme model compression. *arXiv preprint arXiv:2004.07320* (2020). 6

[GCL*21] GUO M.-H., CAI J.-X., LIU Z.-N., MU T.-J., MARTIN R. R., HU S.-M.: Pct: Point cloud transformer. *Computational Visual Media 7* (2021), 187–199. 2

[GKF09] GOLOVINSKIY A., KIM V. G., FUNKHOUSER T.: Shape-based recognition of 3d point clouds in urban environments. In *2009 IEEE 12th International Conference on Computer Vision* (2009), IEEE, pp. 2154–2161. 1

[GKOM18] GUERRERO P., KLEIMAN Y., OVSJANIKOV M., MITRA N. J.: Pcpnet learning local shape properties from raw point clouds. In *Computer graphics forum* (2018), vol. 37, Wiley Online Library, pp. 75–85. 1

[HMD15] HAN S., MAO H., DALLY W. J.: Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149* (2015). 3

[Hor91] HORNIK K.: Approximation capabilities of multilayer feedforward networks. *Neural networks 4*, 2 (1991), 251–257. 2

[JKC*18] JACOB B., KLIGYS S., CHEN B., ZHU M., TANG M., HOWARD A., ADAM H., KALENICHENKO D.: Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 2704–2713. 3

[KMM*19] KALAMKAR D., MUDIGERE D., MELLEMPUDI N., DAS D., BANERJEE K., AVANCHA S., VOOTURI D. T., JAMMALAMADAKA N., HUANG J., YUEN H., ET AL.: A study of bfloat16 for deep learning training. *arXiv preprint arXiv:1905.12322* (2019). 3

[LBS*18] LI Y., BU R., SUN M., WU W., DI X., CHEN B.: Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems 31* (2018). 2

[LKH21] LEE J., KIM D., HAM B.: Network quantization with element-wise gradient scaling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2021), pp. 6448–6457. 3, 6, 7

[LLLB17] LIU C., LI K., LI K., BUYYA R.: A new service mechanism for profit optimizations of a cloud provider and its users. *IEEE Transactions on Cloud Computing 9*, 1 (2017), 14–26. 8

[LNL*19] LIU J., NI B., LI C., YANG J., TIAN Q.: Dynamic points agglomeration for hierarchical point sets learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 7546–7555. 6

[LTL13] LI K., TANG X., LI K.: Energy-efficient stochastic task scheduling on heterogeneous computing systems. *IEEE Transactions on Parallel and Distributed Systems 25*, 11 (2013), 2867–2876. 1

[LUW17] LOUIZOS C., ULLRICH K., WELLING M.: Bayesian compression for deep learning. *Advances in neural information processing systems 30* (2017). 3

[LWH*22] LIU Z., WANG Y., HAN K., MA S., GAO W.: Instance-aware dynamic neural network quantization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2022), pp. 12434–12443. 3

[LYWU18] LIANG M., YANG B., WANG S., URTASUN R.: Deep continuous fusion for multi-sensor 3d object detection. In *Proceedings of the European conference on computer vision (ECCV)* (2018), pp. 641–656. 1

[LZC*21] LIU Z., ZHANG Z., CAO Y., HU H., TONG X.: Group-free 3d object detection via transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 2949–2958. 6

[ML22] MA L., LI J.: Sd-gcn: Saliency-based dilated graph convolution network for pavement crack extraction from 3d point clouds. *International Journal of Applied Earth Observation and Geoinformation 111* (2022), 102836. 4

[MNA*17] MICIKEVICIUS P., NARANG S., ALBEN J., DIAMOS G., ELSEN E., GARCIA D., GINSBURG B., HOUSTON M., KUCHAIEV O., VENKATESH G., ET AL.: Mixed precision training. *arXiv preprint arXiv:1710.03740* (2017). 3

[MQY*22] MA X., QIN C., YOU H., RAN H., FU Y.: Rethinking network design and local geometry in point cloud: A simple residual mlp framework. *arXiv preprint arXiv:2202.07123* (2022). 2

[NTR*20] NEZHADARYA E., TAGHAVI E., RAZANI R., LIU B., LUO J.: Adaptive hierarchical down-sampling for point cloud classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 12956–12964. 5

[Pap23] PAPPALARDO A.: Xilinx/brevitas, 2023. URL: https://doi.org/10.5281/zenodo.3333552, doi:10.5281/zenodo.3333552. 6

[PMJ*16] PAPERNOT N., MCDANIEL P., JHA S., FREDRIKSON M., CELIK Z. B., SWAMI A.: The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)* (2016), IEEE, pp. 372–387. 4

[PWT*22] PANG Y., WANG W., TAY F. E., LIU W., TIAN Y., YUAN L.: Masked autoencoders for point cloud self-supervised learning. In *European conference on computer vision* (2022), Springer, pp. 604–621. 2

[QCZ*20] QIN H., CAI Z., ZHANG M., DING Y., ZHAO H., YI S., LIU X., SU H.: Bipointnet: Binary neural network for point clouds. *arXiv preprint arXiv:2010.05501* (2020). 4

[QSMG17] QI C. R., SU H., MO K., GUIBAS L. J.: 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. 2, 4, 5, 6

[QYSG17] QI C. R., YI L., SU H., GUIBAS L. J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems 30* (2017). 2, 6

[RMB*08] RUSU R. B., MARTON Z. C., BLODOW N., DOLHA M., BEETZ M.: Towards 3d point cloud based object maps for household environments. *Robotics and Autonomous Systems 56*, 11 (2008), 927–941. 1

[SVZ13] SIMONYAN K., VEDALDI A., ZISSERMAN A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034* (2013). 4

[SYX*21] SHU Z., YANG S., XIN S., PANG C., JIN X., KAVAN L., LIU L.: Detecting 3d points of interest using projective neural networks. *IEEE Transactions on Multimedia 24* (2021), 1637–1650. 4

[Tao20] TAO A.: Dgcnn pytorch, 2020. URL: https://github.com/antao97/dgcnn.pytorch. 6

[TFML20] TAILOR S. A., FERNANDEZ-MARQUES J., LANE N. D.: Degree-quant: Quantization-aware training for graph neural networks. *arXiv preprint arXiv:2008.05000* (2020). 3, 4

[UPH*19] UY M. A., PHAM Q.-H., HUA B.-S., NGUYEN D. T., YEUNG S.-K.: Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *International Conference on Computer Vision (ICCV)* (2019). 2, 6

[VSP*17] VASWANI A., SHAZEER N., PARMAR N., USZKOREIT J., JONES L., GOMEZ A. N., KAISER Ł., POLOSUKHIN I.: Attention is all you need. *Advances in neural information processing systems 30* (2017). 2

[WDW*22] WANG L., DONG X., WANG Y., LIU L., AN W., GUO Y.: Learnable lookup table for neural network quantization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2022), pp. 12423–12433. 3, 6, 7

[WLL*19] WANG K., LIU Z., LIN Y., LIN J., HAN S.: Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2019), pp. 8612–8620. 3

[WSK*15] WU Z., SONG S., KHOSLA A., YU F., ZHANG L., TANG X., XIAO J.: 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 1912–1920. 2, 6

[WSL*19] WANG Y., SUN Y., LIU Z., SARMA S. E., BRONSTEIN M. M., SOLOMON J. M.: Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (tog) 38*, 5 (2019), 1–12. 2, 6

[WSM*18] WANG S., SUO S., MA W.-C., POKROVSKY A., URTASUN R.: Deep parametric continuous convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 2589–2597. 1

[WZL*22] WENG T., ZHOU X., LI K., TAN K.-L., LI K.: Distributed approaches to butterfly analysis on large dynamic bipartite graphs. *IEEE Transactions on Parallel and Distributed Systems 34*, 2 (2022), 431–445. 2

[XLC*19] XIAO G., LI K., CHEN Y., HE W., ZOMAYA A. Y., LI T.: Caspmv: A customized and accelerative spmv framework for the sunway taihulight. *IEEE Transactions on Parallel and Distributed Systems 32*, 1 (2019), 131–146. 7

[XSW*20] XU Q., SUN X., WU C.-Y., WANG P., NEUMANN U.: Grid-gcn for fast and scalable point cloud learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 5661–5670. 6

[Yan19] YAN X.: Pointnet/pointnet++ pytorch, 2019. URL: https://github.com/yanx27/Pointnet_Pointnet2_pytorch. 6

[YLS*21] YU H., LI H., SHI H., HUANG T. S., HUA G.: Any-precision deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2021), vol. 35, pp. 10763–10771. 3, 6, 7

[YSLJ20] YANG Z., SUN Y., LIU S., JIA J.: 3dssd: Point-based 3d single stage object detector. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2020), pp. 11040–11048. 6

[YTR*22] YU X., TANG L., RAO Y., HUANG T., ZHOU J., LU J.: Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2022), pp. 19313–19322. 2

[ZBIW19] ZAFRIR O., BOUDOUKH G., IZSAK P., WASSERBLAT M.: Q8bert: Quantized 8bit bert. In *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS)* (2019), IEEE, pp. 36–39. 3

[ZCY*19] ZHENG T., CHEN C., YUAN J., LI B., REN K.: Pointcloud saliency maps. In *Proceedings of the IEEE/CVF international conference on computer vision* (2019), pp. 1598–1606. 4

[ZHW*19] ZHANG K., HAO M., WANG J., DE SILVA C. W., FU C.: Linked dynamic graph cnn: Learning on point cloud via linking hierarchical features. *arXiv preprint arXiv:1904.10014* (2019). 2

[ZJJ*21] ZHAO H., JIANG L., JIA J., TORR P. H., KOLTUN V.: Point transformer. In *Proceedings of the IEEE/CVF international conference on computer vision* (2021), pp. 16259–16268. 2

[ZLZ*24] ZHOU S., LI L., ZHANG X., ZHANG B., BAI S., SUN M., ZHAO Z., LU X., CHU X.: Lidar-ptq: Post-training quantization for point cloud 3d object detection. *arXiv preprint arXiv:2401.15865* (2024). 4

[ZST*18] ZHUANG B., SHEN C., TAN M., LIU L., REID I.: Towards effective low-bitwidth convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 7920–7928. 6

[ZYX*21] ZHONG K., YANG Z., XIAO G., LI X., YANG W., LI K.: An efficient parallel reinforcement learning approach to cross-layer defense mechanism in industrial control systems. *IEEE Transactions on Parallel and Distributed Systems 33*, 11 (2021), 2979–2990. 1