Multi-Task Y-Shaped Graph Neural Network for Point Cloud Learning in Autonomous Driving

Xiaofeng Zou, Kenli Li, *Senior Member, IEEE*, Yangfan Li, Wei Wei, *Senior Member, IEEE*, and Cen Chen

***Abstract*— Point cloud, an efficient 3D object representation, plays an indispensable role in autonomous driving technologies, such as object avoidance, localization, and map building. The analysis of point clouds (e.g., 3D segmentation) is essential to exploit the informative value of point clouds for such applications. The main challenge remains to effectively and completely extract high-level point cloud feature representations. To this end, we present a novel multi-task Y-shaped graph neural network to explore 3D point clouds, referred to as MTYGNN. By extending the conventional U-Net, MTYGNN contains two main branches to simultaneously perform classification and segmentation tasks in point clouds. Meanwhile, the classification prediction is fused together with the semantic features as the scene context to make the segmentation task more accurate. Furthermore, we consider the homoscedastic uncertainty of each task to calculate the weights of multiple loss functions to ensure that tasks do not negatively interfere with each other. The proposed MTYGNN is evaluated on popular point cloud datasets in traffic scenarios. Experimental results demonstrate that our framework outperforms the state-of-the-art baseline methods.**

***Index Terms*— Graph neural networks, multi-task learning, point clouds, Y-shaped architecture.**

I. Introduction

With the continuous growth of market demand for autonomous driving, traffic monitoring, semantic high-definition maps, and smart cities, mobile laser scanning systems have attracted widespread attention from many researchers [1]. The system can effectively collect high-precision and high-density three-dimensional (3D) point cloud data in complex urban road environments, which can help self-driving vehicles achieve localization and obstacle tracking. 3D point cloud data has been broadly used in various domains, including 3D object detection on urban roads, object classification, object tracking [2], semantic segmentation [3], etc. To meet the important demands of intelligent transportation in smart cities, it is crucial to devise the methods that can capture valuable information from point clouds. In this work, we focus on the theoretical and fundamental problem of point cloud segmentation in autonomous driving scenarios.

	Image	Point Cloud
Premutation	Ordered	Orderless
Data Structure	Regular	Irregular
Dimension	2D	3D
Data Type	Discrete	Continuous

Fig. 1. The comparison between 2D image and 3D point cloud. Compared with 2D images, point clouds can provide more rich geometry, shape and scale information, and is not easily affected by changes in light intensity and other objects occlusion.

As an efficient representation of 3D data, point cloud can accurately represent arbitrarily shaped objects and the surrounding environment in 3D space. Fig. 1 presents the comparison between 3D point cloud and 2D image. Compared with 2D images, it has incomparable advantages. Because it can provide rich geometry, shape, and scale information, and is not easily affected by changes in light intensity and other objects occlusion. Thus, 3D point clouds can help self-driving cars reconstruct their 3D environment and better understand the complex traffic environment. However, the disorder and irregularity of point clouds pose a great challenge to research.

With the great breakthrough of convolutional neural networks (CNNs) in 2D images [4], [5], many studies [6]–[8] attempt to explore deep learning approaches to directly analyze and understand point clouds. Qi *et al.* [7] proposed the famous PointNet, which employs multi-layer perceptrons (MLPs) for point-wise feature learning and utilizes symmetric functions (e.g., maximum pooling) to aggregate global features. However, this approach processes each point individually, regardless of the local structure message for the point cloud.

Manuscript received 30 April 2021; revised 7 September 2021 and 6 January 2022; accepted 19 January 2022. Date of publication 17 February 2022; date of current version 8 July 2022. This work was supported in part by the National Key Research and Development Programs of China under Grant 2020YFB2104000, in part by the Cultivation of Shenzhen Excellent Technological and Innovative Talents (Ph.D. Basic Research Started) under Grant RCBS20200714114943014, in part by the Basic Research of Shenzhen Science and Technology Plan under Grant JCYJ20210324123802006, and in part by the National Natural Science Foundation of China under Grant 61902120. The Associate Editor for this article was S. Wan. *(Corresponding author: Kenli Li.)*

Xiaofeng Zou, Kenli Li, and Yangfan Li are with the College of Computer Science and Electronic Engineering, Hunan University, Hunan 410082, China (e-mail: zouxiaofeng@hnu.edu.cn; lkl@hnu.edu.cn; yangfanli@hnu.edu.cn).

Wei Wei is with the School of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China (e-mail: weiwei@xaut.edu.cn).

Cen Chen is with the Institute for Infocomm Research (I2R), Singapore 138632 (e-mail: chenc@i2r.a-star.edu.sg).

Digital Object Identifier 10.1109/TITS.2022.3150155

To address this issue, PointNet++ [8] developed a hierarchical architecture to capture multi-scale features. Nevertheless, the dependency relationships between the points are still not considered.

Recently, the graph neural network (GNNs) [9]–[11] have applied CNNs to irregular graph data, which has attracted widespread attention. The key idea is to aggregate the features from neighbor nodes according to the graph topology, thereby can mine the relationships and interactions among graph nodes. Since sparse and irregular point cloud data can be naturally represented as graph data, some recent works have developed GNN-based methods [12]–[15] for point clouds. GNN models require graph structure and node features as input. To be suitable for unstructured point cloud data, these methods utilize the k-nearest neighbor (k-NN) algorithm to build the graph structure, and then the generated graph structure and the coordinates of the point cloud are fed into graph convolution layer for feature learning. Despite these methods have significantly improved performance, there are still some issues worth considering: (1) *These methods generally do not consider multi-task learning and only focus on a single task.* Multi-task learning [16] can explore the commonalities among tasks by joint learning to obtain better generalization. (2) *The graph convolutional layer in the point cloud be further improved.* Since the k-NN graph only considers the positional relationship between points, ignoring the potential structural information. (3) *It is necessary to design an efficient down-sampling method for disordered point clouds.* Existing down-sampling methods [8], [17], [18] do not study the importance of points in the point cloud, which may cause some important points to be deleted, thereby affecting performance.

To address these issues, we propose a novel multi-task Y-shaped graph neural network to process point clouds, referred to as MTYGNN. By extending the traditional U-shaped framework [19], MTYGNN performs classification and segmentation tasks in the point cloud simultaneously. It consists of three key parts: a top-down shared encoder, classification branch, and segmentation branch. We integrate classification prediction as the scene context into the features extracted by segmentation branch for better performance. To extract the potential structural information in the point cloud, we develop a graph attention convolutional (GAConv) layer, which utilizes the attention mechanism [20] to dynamically assign appropriate attention weights to different neighbors. We further develop a novel graph adaptive pooling (GAPool) layer, which adaptively retains important points based on the potential feature and structure message of the point cloud. To support the up-sampling operation in the point cloud, we adopt the graph unpooling (PUnpool) layer proposed in [11], which effectively recovers the reduced graph to raw size. Finally, we consider the homoscedastic uncertainty [21] of each task to compute the weights of multiple loss functions to ensure that tasks do not negatively interfere with each other in multi-task learning.

To our knowledge, this is the first multi-task graph neural network designed for point clouds, which can explore commonalities among multiple point cloud-related tasks. The main contributions of this work are summarized as follows:

- We propose a new MYTGNN, which can exploit the commonality between multiple tasks to improve the generalization ability of the model.
- We develop a GAConv and a GAPool to extract better feature representations for point clouds. The proposed GAConv utilizes attention mechanisms to explore the underlying structural relationships of the point cloud, while GAPool considers both feature and topological information of the point cloud to achieve the pooling operation.
- We explore a weighted loss function to ensure that tasks do not interfere with each other.
- Our MTYGNN outperforms various state-of-the-art methods on two common point cloud datasets in traffic scenarios, which effectively proves the excellence of MTYGNN.

II. Related Work

A. Deep Learning on Point Clouds

Recently, analyzing and understanding point clouds has become a hot research topic, and a series of deep learning-based methods have been proposed. These studies can be divided into two main categories: CNN-based approaches and Graph-based approaches. Moreover, we review some down-sampling approaches in graphs.

1) CNN-Based Approaches: PointNet [7] was a pioneering work to directly employ deep learning models to original point clouds. It adopted MLPs to extract the point-wise feature, and then the symmetric functions were used to summarize the global features. Nevertheless, this approach processes each point individually, regardless of the local structure message for point clouds. To this end, Qi *et al.* [8] developed Pointnet++, which extracts multi-scale local features by constructing a hierarchical framework. Since PointNet and PointNet++ are simple and efficient, many researchers [22]–[24] develop more powerful models based on these two models. However, the series of models ignore the topology relationships between points.

2) Graph-Based Approaches: The vigorous development of graph neural networks (GNNs) [9], [10], [25] has brought great opportunities for irregular point cloud feature learning. Specifically, GNNs aim to extend the neural networks to irregular graph data. The key idea is to aggregate the features from adjacent nodes based on the graph structure. While sparse and irregular point cloud data can be naturally expressed as graph data, some recent work [12]–[15] has developed GNN-based methods to explore point clouds. Since the GNN model requires graph structure and node features as input, these methods first generate graph edges for unstructured point cloud data. Te *et al.* designed RGCNN [12], which dynamically constructs graphs by exploring the similarity of feature spaces. Wang *et al.* [13] developed DGCNN, which utilizes EdgeConv to dynamically construct k-NN graphs in each layer of the network. LDGCNN [14] further improved DGCNN, which connects hierarchical features from multi-levels in a residual manner to improve performance Nevertheless, these above methods generally do not consider multi-task learning and only focus on a single task.

3) Down-Sampling Methods on Point Clouds: The down-sampling operation can generalize the hierarchical features and expand the receiving field to obtain better generalization. Some researches [8], [17], [18], [26] have started to extend the down-sampling operation to point cloud data processing. Pointnet++ [8] employed farthest point sampling (FPS) to down-sample point clouds. However, this sampling method only considers the spatial location of points in the input point cloud, not their feature representation. Alternative research [17] utilized graph-based max-pooling to perform the downsampling operation. The limitation of this method is that it does not ensure that the points with significant contributions are retained during sampling, which can bring troubles to downstream feature learning. Following this line, [18] further proposed a sampling strategy based on the deep learning approach, which utilizes a neural network to generate a reduced point cloud from the original dataset. However, this method is not placement-invariant. It generates a new point set during down-sampling, which changes the order of the input point cloud and makes it difficult to track each selected point node, resulting in the reduced point cloud that cannot be restored to its original size.

B. Multi-Task Learning

Multi-task learning (MTL) [16] is an effective learning paradigm in deep learning, which utilizes the commonality among tasks to improve the generalization capability of the model. Specifically, it mines useful information between tasks by extracting shared features with sufficient expressive ability. Recently, MTL-based models have been broadly used in many domains, including object detection, image segmentation, and image classification. However, MTL on graph neural networks has not attracted much attention, and only few studies [27], [28] mainly focus on the multi-head architecture. However, the research [27] found that direct training of the multi-head architecture model resulted in a performance loss for a single task. Thus, we design a novel Y-shaped framework for multi-tasking learning, which can learn generalize node embedding across tasks.

III. Methodology

In this section, we introduce the definition of multi-task learning problem in point clouds, and describe the proposed MTYGNN in detail.

A. Problem Formulation

As depicted in Fig. 2, the point cloud P is composed of a set of 3D points: $\{p_i = (x_i, y_i, z_i) \,|i = 1, 2, \ldots, n\}$, where n is the number of points, and p_i is a point in the point cloud and consists of 3D coordinates (x_i, y_i, z_i). We utilize $X \in \mathbb{R}^{n\times 3}$ to represent the feature matrix of P.

1) Point Cloud Classification: Given a point cloud P, we classify the class of the entire point clouds. Therefore, we need to develop a mapping function f_c that maps point cloud features to probability distributions over each category:

$$f_c\left(\{p_1, p_2, \ldots, p_n\}\right) \rightarrow N_c, \tag{1}$$

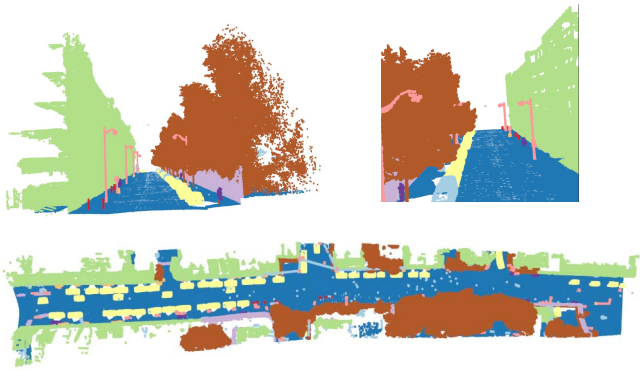

Fig. 2. Illustration of point clouds in intelligent transportation.

where N_c is the object category of point cloud (e.g., car, motor, airplane).

2) Point Cloud Segmentation: In this setting, each point p_i should be classified as a specific semantic category. Thus, a segmentation function f_s needs to be developed to compute the probability distribution of each category on each point p_i:

$$\{f_s\left(p_i\right) |i = 1, 2, \ldots, n\} \rightarrow N_s, \tag{2}$$

where N_s is the semantic labels for each points.

In this work, we design a graph neural network based on multi-task learning to simultaneously perform point cloud classification and segmentation tasks. These two problems are usually solved separately in previous work. However, we have observed that the object category and semantic labels are interdependent in the point cloud. For instance, the category of the entire point cloud can serve as a global context to help identify the category of each point. Therefore, it is desirable to couple point cloud classification and point cloud segmentation into a single task.

B. Overall Architecture of MTYGNN

As illustrated in Fig. 3, MTYGNN explores a novel Y-shaped architecture, which includes a shared encoder for feature extraction and two parallel branches for classification and segmentation tasks, respectively. It takes the original point cloud as input and extracts hierarchical feature information through a shared encoder. The two parallel branches then obtain the learned feature representations and perform their subsequent predictions separately.

The shared encoder aims to project the point clouds into a new high-dimensional feature space to encode more local information than the raw space. Inspired by the classic U-shaped model, the shared encoder part is a top-down encoder, which includes several encoding modules, each consisting of a proposed GAConv layer and a proposed GAPool layer. GAConv layer employs the attention mechanism to mine the hidden topological information between points in the point cloud, and exploits neighborhood aggregation to capture local information. GAPool layer adaptively performs graph coarsening to summarize hierarchical features for the point clouds.

In the classification branch, we employ the feature extraction module consistent with the encoder to further learn

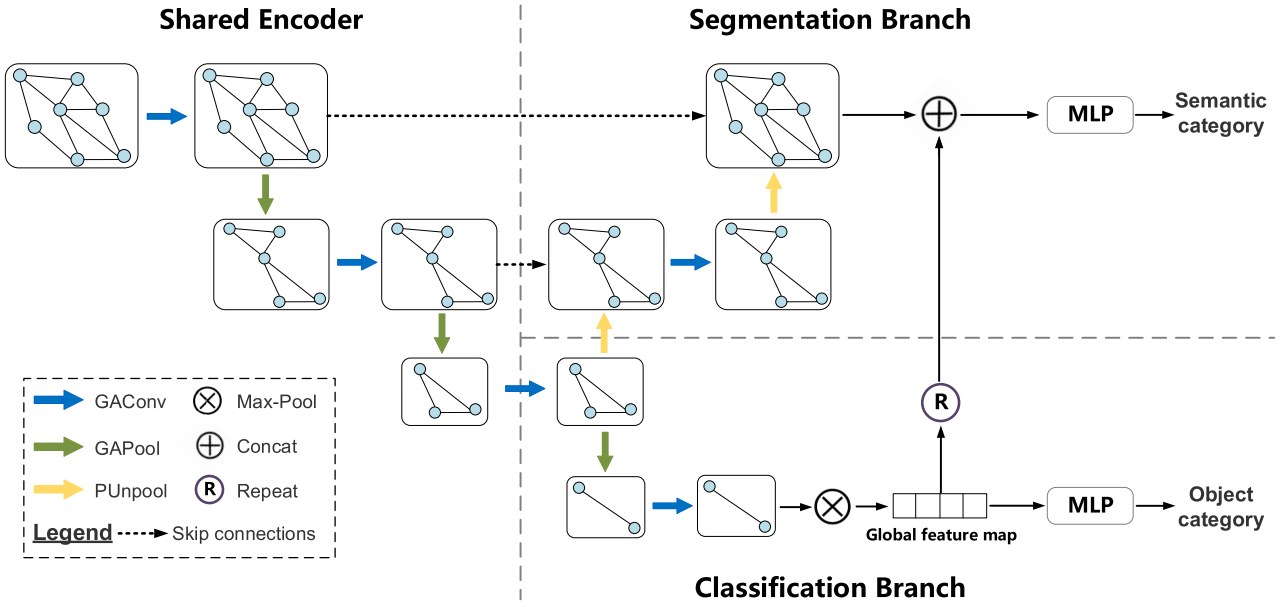

Fig. 3. Overall framework of the proposed MTYGNN. It consists of three main parts: a shared encoder, a classification branch and a segmentation branch.

the point-wise feature representation, and then adopt the max-pooling operation to extract effective global features, and finally utilize the MLPs to perform the final point cloud classification. To maximize the learned multi-scale features, we combine hierarchical features to obtain better representation ability. In specific, we apply a max-pooling operation after each encoding module to acquire a global feature, and then combine these hierarchical features for the final classification.

In the segmentation branch, we adopt a bottom-up decoder to restore the original size of the point cloud to achieve point-wise prediction. The decoder consists of several decoding modules, the number of which is the same as the encoding module in shared encoder. Each decoding module consists of a GAConv layer and a PUnpool layer. GAConv layer is responsible for extracting local information, while the PUnpool layer is responsible for recovering the reduced features. Following the traditional U-shaped network, we add skip-connections between encoding and decoding modules of the same level, which can effectively combine low-level node features and high-level features for more expressive features. Finally, the global features extracted by the classification branch will be incorporated into the feature representation extracted by the decoder as the scene context information to improve the representation capability of the model.

MTYGNN is an extension of the traditional U-shaped model (e.g., Graph U-Net [11]) to jointly perform segmentation and classification tasks. The main distinctions between MTYGNN and Graph U-Net in the following ways: (i) Different from Graph U-Net that only supports a single task, MTYGNN adopts a Y-shaped model framework that supports multi-task learning. It can exploit the commonalities among multiple point cloud-related tasks to improve performance. (ii) GAConv proposed in MTYGNN is different from the GCN. GAConv assigns appropriate attention weights to different neighbor points through the attention mechanism, and

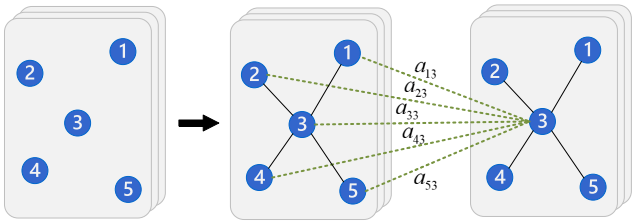

Fig. 4. Illustration of GAConv on a subgraph of a point cloud.

selectively gathers the most relevant neighbor information to dynamically capture local information. (iii) The differences between GAPool and gPool in Graph U-Net are significant. gPool only utilizes feature messages of the point clouds during graph coarsening, while GAPool adaptively retains important points based on feature information and structural messages.

C. Graph Attention Convolution for Point Clouds

Since the previous graph neural networks [9], [10] require graph structure and feature matrix as input, while point cloud data only contains geometric features, and graph structure is unknown, these approaches cannot be directly extended to point clouds. To resolve this issue, we specially design a graph attention convolutional (GAConv) layer for unstructured point clouds, as depicted in Fig. 4. It mainly includes two processes: (1) graph construction, (2) graph feature aggregation.

1) Graph Construction: An appropriate graph structure has a great impact on the quality of feature representation learning, thus we need to choose a suitable graph construction method. A simple and efficient method involves employing the k-nearest neighbor (k-NN) algorithm to build a directed graph $\mathcal{G}^l$. However, the k-NN algorithm only considers the positional relationship between points, which may cause the loss of hidden structural information. To optimize the constructed k-NN graph $\mathcal{G}^l$, we adopt the attention mechanism to explore

Algorithm 1 The Computing Process of MTYGNN

Input: Raw point clouds features X^0, shared encoder network depth L_1, classification branch network depth L_2.
Output: Object category label N_c, semantic labels N_s.

Shared encoder:
$X_e^0 = X^0$; # X_e is the feature representation extracted by the shared encoder.
for $l = 0$ to L_1 **do**
  Feature extraction $X_e^l + 1$ via GAconv;
  Pooling operation via GAPool
  $l = l + 1$;
end for

Classification branch:
$X_c^0 = X_e^{L_1}$; # X_c is the feature representation extracted by the classification branch.
for $l = 0$ to L_2 **do**
  Feature extraction X_c^l via GAconv;
  Pooling operation via GAPool
  $l = l + 1$;
end for
compute the category label N_c by MLP

Segmentation branch:
$X_s^0 = X_e^{L_1}$; # X_s is the feature representation extracted by the segmentation branch.
for $l = 0$ to L_1 **do**
  Up-sampling operation via PUnpool
  skip-connections;
  Feature extraction X_s^l for point clouds via GAconv;
  $l = l + 1$;
end for
compute the semantic category N_s by MLP;
return N_c,N_s

the pairwise dependencies in the feature space. Specifically, the attention mechanism is used to adjust the weights of the graph edges in $\mathcal{G}^l$:

$$e_{ij}^l = \alpha\left(wx_i^l, wx_j^l, w\left(x_i^l - x_j^l\right)\right), \quad j \in N(i), \tag{3}$$

where the sharing attention mechanism $\alpha(\cdot)$ is a single-layer neural network, x_i^l and x_j^l mean the feature vectors of point i and j in layer l, $j \in \mathcal{N}(i)$ is the neighborhood of point i. Different from GAT, we extra incorporate feature differences among node pairs $\left(x_i^l - x_j^l\right)$ in the attention mechanism, which can help the model pay more attention to similar neighbors. e_{ij} indicates the attentional weight between point i and j. w is the trainable weight parameter. After obtaining the pairwise relationship between the points, we normalized them with softmax function to get the final edge weights α_{ij}^l, i.e.,

$$a_{ij}^l = \text{softmax}\left(e_{ij}^l\right) = \frac{\exp\left(e_{ij}^l\right)}{\sum_{k\in N(i)} \exp\left(e_{ik}^l\right)}. \tag{4}$$

2) Graph Feature Aggregation: After obtaining the graph structure, we utilize the neighborhood aggregation to extract the local graph features. Specifically, we aggregate neighbor information based on the attention values to form a new node representation. the feature aggregation function of point i is defined as follows:

$$x_i^{l+1} = \sigma\left(\sum_{j\in N(i)} a_{ij}^l W x_j\right), \tag{5}$$

where σ is the activation function, x_i^{l+1} is the updated node feature of point i. Since the feature aggregation function operates the same for all points, the feature representation for each point can be updated in parallel.

Although both our proposed GAConv and GAT utilize the attention mechanism to learn better representations, our GAConv has significant differences: (1) GAT is not directly applicable to point cloud data with unknown structure, while GAConv is specifically designed for the properties of point cloud data. (2) GAConv incorporates feature differences to learn attention weights, which can extract more precise node interaction relationships.

D. Graph Adaptive Pooling for Point Clouds

Recently, graph pooling layers [11], [29] have received widespread attention. These methods are used to reduce the size of the graph, and summarize abstract features to achieve better generalization. Nevertheless, the current research on point cloud pooling is not thorough enough. The previous approaches [8], [17], [18] are mostly static and sensitive to abnormal points. Since they only sample the points in the input point clouds based on their spatial positions, instead of based on their feature representations. Furthermore, these methods are not permutation-invariant. Since they summarize the original nodes into a new set of coarsened nodes during the down-sampling process, rather than selecting a subset of the input points. In this manner, it is difficult to restore the reduced point cloud to its original size, which in turn cannot be extended to point cloud segmentation tasks.

To solve the mentioned issues, we explore a novel graph adaptive and permutation-invariant pooling (GAPool) layer for point clouds, which exploits the feature representation and structure information of the point cloud to adaptively retain important points for down-sampling. In particular, we transform the pooling problem into a node sampling problem. The importance of points is learned by considering the graph topology and feature representation jointly, and retains K most important points to generate a new graph. Fig. 5 depicts the process of GAPool, which is mainly divided into a topology-aware attention phase and a node sampling phase.

1) Topology-Aware Attention: The attention mechanism [20] can drive the model to focus on the most relevant parts of the task. Inspired by this, we develop a topology-aware attention scheme to measure the importance of points. Unlike previous methods [11], our approach explicitly combines graph topology and node features. To be specific, we gather the neighborhood information for each point depending on

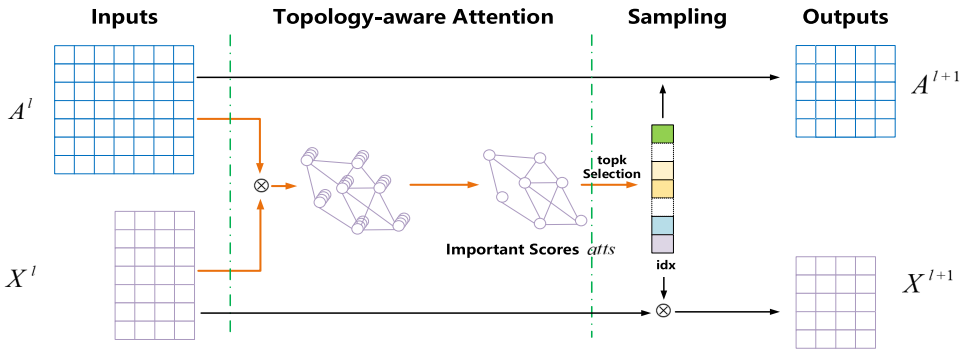

Fig. 5. Illustration of the graph adaptive pooling layer (GAPool). There are two main phases in GAPool: : topology-aware attention and node sampling. Topology-aware attention aims to learn the importance of points $atts$ by exploiting graph topology A^l and node features X^l. Then K points with highest scores are selected for the down-sampling operation.

the graph structure to generate a new feature representation with structural information. Self-attention [30] is then used to explore the importance of each point in the whole graph. Given the graph $G = \left(X^l, A^l\right)$ with the feature matrix X^l and adjacency matrix A^l, the importance of points is obtained as:

$$\begin{aligned} \hat{X} &= \hat{D}^{-1}\hat{A}^l X, \\ atts &= \text{softmax}\left(\sigma\left(w_s\hat{X}\right)\right), \end{aligned} \tag{6}$$

where $atts \in \mathbb{R}^{n\times 1}$ is the importance score of points, $\hat{A}^l = A^l + I$, $\hat{D}$ is the degree matrix of $\hat{A}^l$, and $1_n = [1, \ldots, 1]^{\mathrm{T}}$, $w_s \in \mathbb{R}^{h\times 1}$ is the learnable parameter.

2) Graph Node Sampling: After obtaining the importance scores of points, we perform node sampling to keep points with higher scores. The graph nodes are re-ranked according to the node information scores $atts$, and K important nodes are sampled to form new node features X^{l+1} and graph structure A^{l+1}.

$$\begin{aligned} \text{idx} &= \text{topk}\left(atts, \text{K}\right), \\ A^{l+1} &= A^l\left(\text{idx}, \text{idx}\right), \\ X^{l+1} &= X^l\left(\text{idx}, :\right), \end{aligned} \tag{7}$$

where K is a hyper-parameter to determine the number of points reserved in the new graph, $\text{topk}\left(atts, \text{K}\right)$ function performs node sampling and it returns the indexes of the K maximum values in $atts$. $A^l\left(\text{idx}, \text{idx}\right)$ and $X^l\left(\text{idx}, :\right)$ perform the column or row extraction operation based on the indexes idx. With indexes idx, we obtain the new adjacency matrix $A^{l+1} \in \mathbb{R}^{\text{K}\times\text{K}}$ and new feature matrix $X^{l+1} \in \mathbb{R}^{k\times F}$.

In this manner, we can adaptively retain the points with significant contributions to achieve the down-sampling operation. Moreover, our proposed GAPool is permutation-invariant. This is because GAPool utilizes the features of the point cloud for importance ranking and selects directly among the set of input point based on the importance without generating new points. It does not change the permutation of the input points, i.e., order-independent, which allows subsequent upsampling operations to easily keep tracks of each selected node.

E. Point Cloud Unpooling (PUnpool) Layer

The unpooling operation can restore the reduced feature matrix to the original size. However, the previous unpooling operation is not available due to the irregularity of the point clouds. In MTYGNN, we developed a graph pooling (PUnpool) layer to perform up-sampling operation, which is the inverse operation of GAPool. During GAPool, idx preserves the indexes of the reserved node in the raw feature matrix. In PUnpool, we put the selected nodes back to their original places in the graph based on idx, and produce a new feature matrix X^{l+1}. Formally, PUnpool can be represented as

$$X^{l+1} = D\left(0_{n\times h}, X^l, \text{idx}\right), \tag{8}$$

where $0_{n\times h}$ is the initially empty matrix, and D function employs the indexes idx to fill the selected node features X^l in GAPool into $0_{n\times h}$.

F. Multi-Task Loss Function

There is a common problem in multi-task learning: it needs to optimize the proposed model for multiple objectives. In our work, the proposed MTYGNN uses a Y-shaped model framework to support multitask learning. It contains two branches to perform two tasks: predict the object label of the point cloud, and perform pointwise forecasts of 3D points in the point cloud. To avoid mutual interference among subtasks in multitask learning, we need a suitable multitask loss function to ensure the quality of model training. Here, we denote the classification and segmentation objectives as $\mathcal{L}_{clc}$ and $\mathcal{L}_{seg}$, respectively. A simple idea for defining a multi-task objective is to perform a weighted linear combination of the losses for two tasks:

$$\mathcal{L}_{total} = w_1\mathcal{L}_{clc} + w_2\mathcal{L}_{seg}, \tag{9}$$

where w_1 and w_2 are the task weight parameters, which can normalize the gradient gradients of different tasks to a common scale. The appropriate weight values directly affect the quality of model training. If the correct balance between tasks is not found, the task gradient will conflict, thereby affecting the model performance. A naive approach is to assign weights to the two tasks in the manual manner. However, this approach is heavily dependent on experience and the search for the optimal weights is tedious and expensive. Therefore, it is necessary to find an efficient way to automatically learn the optimal weights.

In this work, we adopt the homoscedastic uncertainty [21] of each task to determine the weight of different task loss functions. Homoscedastic (task-dependent) uncertainty [21] can describe the relative confidence between different tasks, which is independent of the model input data and is not the prediction of model. Therefore, it can be used to design the loss weight terms for different tasks. Here, we apply homoscedastic uncertainty to automatically learn the weights before the classification branch and the segmentation branch. The loss function is defined as follows:

$$L = \frac{1}{2\sigma_1^2} L_{clc} + \frac{1}{2\sigma_2^2} L_{seg} + \log \sigma_1 \sigma_2, \tag{10}$$

where σ_1 and σ_2 are two learned parameters, which represent the homoscedastic uncertainty of the segmentation task and the classification task, respectively. We automatically update σ_1, σ_2 through the backpropagation of the loss function Equ. 10. From Equ. 10, we can observe that the loss of each task is inversely proportional to σ_1, σ_2, so that tasks with smaller uncertainty are assigned larger weights. In addition, the loss of each task is constrained by the uncertainty regularization terms $\log \sigma_i$, which effectively ensures that the model does not predict infinite uncertainty (i.e., zero loss).

G. Complexity Analysis

We first analyze the complexity of each operation in MTYGNN. Given a point cloud with n points, each point contains f features. (i) The computational complexity of GAConv is $O\left(nf + nff' + \xi f'\right)$, where ξ is the set of edges and f' is the hidden feature channel. Compared with GAT, GAConv increases the computational cost of graph construction $O\left(nf\right)$. However, with $\xi \gg n \gg f \approx f'$, the time complexity of GAConv is similar with the general GNNs (e.g., GCN, GAT). (ii) The cost of GAPool is $O\left(\xi f + n\right)$. Since GAPool can effectively reduce the size of the input graph, which allows the computational complexity of the downstream GAConv layers to greatly decrease. After down-sampling through multiple GAPool layers, subsequent GAConv can even be ignored. (iii) The time complexity of PUnpool is $O\left(n + \xi\right)$.

Compared with the existing GNN-based model [13], [15], although MTYGNN introduces an extra classification branch, it does not increase too much computational complexity. The reason is that the down-sampling operation can reduce the complexity of GAConv which is computationally expensive. In particular, the increased computational cost of the classification branch is negligible after multiple down-sampling.

IV. Experiments

In this section, we evaluate our MTYGNN on an outdoor dataset in a large-scale urban environment: Paris-Lille-3D. To further test the scalability and robustness of MTYGNN, we utilize the commonly used dataset, ShapeNet.

A. Paris-Lille-3D

1) Dataset: **Paris-Lille-3D**[1] **[31]** is traffic point cloud data collected from two largest cities in France, Paris, and Lille, by a mobile laser scanning system. In particular, the dataset includes 2 kilometers of urban roads, totaling more than 140 million points. This completely hand-labeled data set contains 50 categories. We follow the guideline in [31] to map the original categories to 9 semantic categories: *Ground, Buildings, Poles, Bollards, Trash Cans, Barriers, Pedestrians, Cars, and Natural*, for a total of 2479 object instances. To satisfy the multi-task setting, we further divide the semantic categories into two classes: foreground and background objects. The details of these staging classes are shown in Table I.

TABLE I
Two Levels of Semantic Classes for the Paris-Lille-3D Data Used in Our Experiments

2-category	Foreground	Background
9-category	Pedestrians, Cars, Trash Cans	Ground, Buildings, Poles, Bollards, Barriers, Natural

2) Baselines: To evaluate the effectiveness of MTYGNN, we compare with some state-of-the-art methods, including: PointNet [7], Pointnet++ [8], DGCNN [13], HDGCN [32], MS-PCNN [3]. Furthermore, we develop a variant to further analyze MTYGNN: $\mathbf{MTYGNN}_{\text{single}}$, which removes the classification branch and performs only the segmentation task.

3) Implementation Details: Our MTYGNN is implemented in PyTorch and trained on 4 NVIDIA Tesla P100 GPUs. We adopt Adam optimizer to optimize the model, the learning rate is 0.01, and the batch size is 16. For the shared encoder, we stacked two encoding modules. We set the nearest neighbor k in the GAConv layer to 20, and set the pooling rate in GAPool to 0.5. The dimension of hidden embedding and dropout rate are 128, 0.5, respectively. In the segmentation branch, we stack two decoding modules to restore the original point cloud size. For the classification branch, we use the same settings as the segmentation branch to support the scene classification. Finally, we feed the features learned by the two branches into an MLP layer for the final prediction scores. To facilitate comparison, we apply the training/validation/test setting consistent with [3].

4) Evaluation Metrics: We utilize two metrics to quantitatively evaluate performance, including the mean intersection over union (mIoU) and the mean accuracy (mAcc).

5) Performance Study: Table II presents the results of our MTYGNN compared with other baselines on the Paris-Lille-3D dataset. Since the baseline methods are all single-task frameworks, the mean accuracy (mAcc) of these methods is not available. From the table, we can observe the following points: (i) We can easily observe that the proposed MTYGNN and the variant $\text{MTYGNN}_{\text{single}}$ have superior performance to all baseline methods. In particular, the mIoU of MTYGNN (75.7%) is 5.2% higher than that of MS-PCNN. These results verify the superiority of our proposed MTYGNN. (ii) The variant $\text{MTYGNN}_{\text{single}}$ outperforms all baseline methods, which effectively indicates that the proposed GAConv and GAPool can capture better feature representations for point clouds. And MTYGNN further outperforms $\text{MTYGNN}_{\text{single}}$. This demonstrates that MTYGNN

[1] https://npm3d.fr/paris-lille-3d

TABLE II
EXPERIMENTAL RESULTS ON PARIS-LILLE-3D BENCHMARK

Baseline	mIoU	Ground	Buildings	Poles	Bollards	Trash Cans	Barriers	Pedestrians	Cars	Natural	mAcc	Foreground	Background
PointNet [7]	14.6	97.3	90.4	22.9	8.7	3.2	2.5	24.3	71.9	26.3	-	-	-
PointNet++ [8]	32.0	96.6	78.6	15.2	4.3	1.2	0	19.3	46.3	26.1	-	-	-
DGCNN [13]	52.9	98.3	93.1	52.9	36.1	19.5	15.0	16.6	88.6	56.5	-	-	-
HDGCN [32]	68.3	99.4	93.0	67.7	75.7	25.7	44.7	37.1	81.9	89.6	-	-	-
MS-PCNN [3]	70.5	98.1	**95.4**	57.6	64.6	**63.0**	34.1	**57.7**	**95.2**	68.3	-	-	-
MTYGNN$_{\text{single}}$	73.4	98.5	93.6	68.6	**88.7**	45.2	42.9	51.8	88.1	82.6	-	-	-
MTYGNN	**75.7**	**99.4**	94.8	**70.9**	88.2	48.6	**49.3**	54.7	90.1	**84.4**	**95.1**	**95.3**	**94.8**

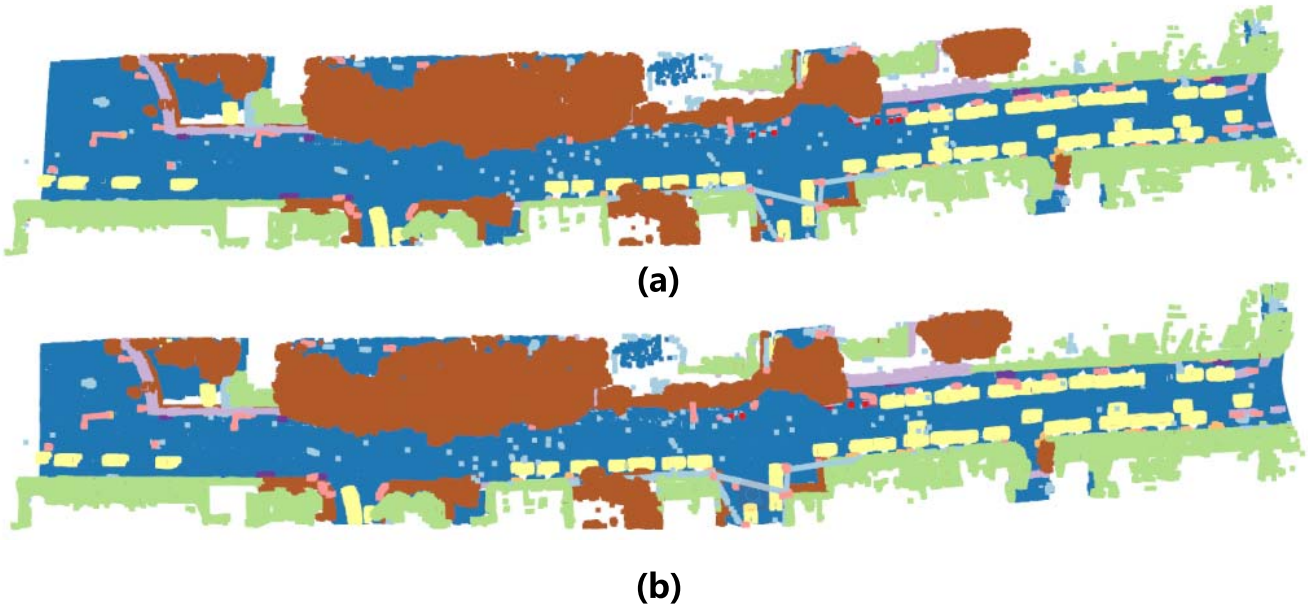

Fig. 6. Visualization results on Paris-Lille-3D dataset. (a) Ground truth. (b) Our MTYGNN.

can explore the commonality between two tasks to improve the model performance through multi-task learning. Fig. 6 presents the segmentation results of MTYGNN in detail.

B. SemanticKITTI

1) Dataset: **SemanticKITTI**[2] **[33]** is a challenging large-scale driving scene point cloud dataset. It consists of 22 sequences, including 43, 552 scans. Following the setting in [26], we utilize sequences 00 to 10 as the training set (where sequence 08 is used as the validation set) and sequences 11 to 21 as the test set. To meet the multi-task setting, we split the 19 valid categories into 6 classes based on [33]: *Ground, Structure, Vehicle, Nature, Human, Object*. The detailed information is presented in Table III.

2) Baselines: We compare MTYGNN with some baselines to validate its superiority, including: PointNet [7], PointNet++ [8], SPLATNet [34], PointASNL [26] which is an end-to-end point cloud processing network based on down-sampling operations, PointConv [35].

3) Implementation Details: For SemanticKITTI, we utilize a similar experimental setup as Paris-Lille-3D. We still use mIoU and mAcc as metrics, where mIoU is used to evaluate the performance of the segmentation branch and mAcc is used to evaluate the performance of the classification branch.

4) Performance Study: The experimental results on the SemanticKITTI dataset are presented in Table IV. From the table, we can observe that our MTYGNN achieves the best performance compared to other single-task baselines. Meanwhile, MTYGNN also outperforms the baseline approach in most categories. It indicates that adopting the multi-task learning framework can explore the commonalities among multiple tasks to learn a better representation for the point cloud. Moreover, the variant MTYGNN$_{\text{single}}$ outperforms PointASNL. It demonstrates the superiority of our GAPool developed for point clouds, which retains points with significant contributions by fully considering the structure information and feature information of the point cloud.

C. ShapeNet

1) Datasets: **ShapeNet**[3] **[36]** is a common 3D object segmentation benchmark. It contains 16681 objects in 16 categories, with a total of 50 parts (each category contains 2-6

[2]http://www.semantic-kitti.org/dataset.html

[3]https://shapenet.cs.stanford.edu/media/

TABLE III
TWO LEVELS OF SEMANTIC CLASSES FOR THE SEMANTICKITTI DATA USED IN OUR EXPERIMENTS

6-category	Ground	Structure	Vehicle	Nature	Human	Object
19-category	road,sidewalk, parking, other-ground	building	car, truck, bicycle, motorcycle, other-vehicle	vegetation trunk terrain	person bicyclist motorcyclist	fence pole traffic sign

TABLE IV
EXPERIMENTAL RESULTS ON SEMANTICKITTI BENCHMARK

Baseline	mIoU	road	sidewalk	parking	other-ground	building	car	truck	bicycle	motorcycle	other-vehicle	vegetation	trunk	terrain	person	bicyclist	motorcyclist	fence	pole	traffic sign	mAcc	Ground	Structure	Vehicle	Nature	Human	Object
SPLATNet [13]	18.4	64.6	39.1	0.4	0.0	58.3	58.2	0.0	0.0	0.0	0.0	71.1	9.9	19.3	0.0	0.0	0.0	23.1	5.6	0.0	-	-	-	-	-	-	-
PointNet++ [8]	20.1	72.0	41.8	18.7	5.6	62.3	53.7	0.9	1.9	0.2	0.2	46.5	13.8	30.0	0.9	1.0	0.0	16.9	6.0	8.9	-	-	-	-	-	-	-
PointASNL [32]	46.8	87.4	74.3	24.3	1.8	83.1	87.9	39.0	0.0	25.1	29.2	84.1	52.2	**70.6**	34.2	**57.6**	0.0	43.9	**57.8**	36.9	-	-	-	-	-	-	-
PointConv [3]	51.2	88.9	68.4	58.9	19.7	84.6	**93.1**	**37.8**	20.7	22.9	38.1	79.9	**62.8**	60.7	46.2	39.1	5.5	52.0	48.1	44.7	-	-	-	-	-	-	-
MTYGNN$_{\text{single}}$	52.4	**91.2**	76.5	**64.8**	26.6	**85.8**	92.3	26.5	23.1	**27.5**	24.0	**84.8**	54.8	63.3	42.8	42.2	6.1	55.5	45.9	51.2	-	-	-	-	-	-	-
MTYGNN	**53.6**	90.9	**78.1**	62.3	**27.8**	85.3	91.2	36.1	**26.0**	25.5	**38.7**	82.2	59.9	64.3	**47.5**	48.9	**7.2**	**56.0**	48.2	**53.1**	**91.5**	**91.7**	**93.8**	**92.7**	**90.4**	**88.2**	**91.2**

TABLE V
EXPERIMENTAL RESULTS ON SHAPENET DATASET

Baseline	mIoU (%)	OA (%)
PointNet [7]	83.7	-
PointNet++ [8]	85.1	-
RGCNN [12]	84.3	-
DGCNN [13]	85.1	-
LDGCNN [14]	85.1	-
MTYGNN$_{\text{single}}$	85.4	-
MTYGNN	**85.7**	**99.6**

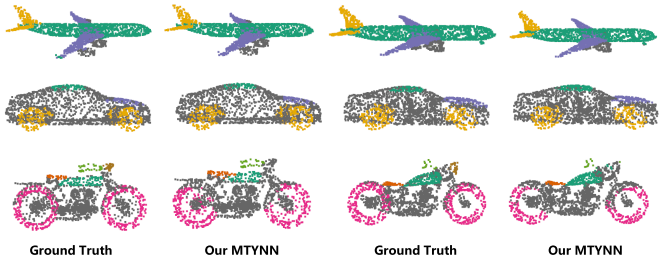

Fig. 7. Visualization results on ShapeNet dataset.

parts). For reasonable comparison, we apply the experimental settings same as [13].

2) Implementation Details: In the segmentation branch, we stack two decoding modules to restore the original point cloud size. For the classification branch, we use two encoding modules to further extract features. Finally, we feed the features learned by the two branches into an MLP layer for the final prediction scores. To facilitate comparison, we apply the training/validation/test setting consistent with [13].

3) Evaluation Metrics: We utilize two metrics to quantitatively evaluate performance, including the overall accuracy (OA) and the intersection over union (IoU) of each category.

4) Performance Study: Table V and Table VI demonstrate the comparison results of the proposed MTYGNN with other baseline methods. It can be seen from Table VI that our proposed MTYGNN achieves the best performance. Meanwhile, MTYGNN also outperforms other baseline methods in most categories. Fig. 7 presents the segmentation results of MTYGNN on ShapeNet dataset in detail.

D. Ablation Studies

1) Effectiveness of the Y-Shaped Architecture: We explore some variants to evaluate the effectiveness of the Y-shaped Architecture: MTYGNN$_{\text{MH}}$ that applys the multi-head architecture consistent with [27]. Table VII shows the experimental results on Paris-Lille-3D and Shapenet datasets. The results show that MTYGNN outperforms MTYGNN$_{\text{MH}}$ in all evaluation metrics. It demonstrates that that the Y-shaped Architecture can hierarchically capture the multi-scale local features.

2) Effects of GAConv: We explore two variants to evaluate the effects of GAConv. (1) DGCNN$_{\text{GAConv}}$ denotes utilizing the proposed GAConv instead of EdgeConv in the DGCNN [13]. (2) MTYGNN$_{\text{EdgeConv}}$ indicates that EdgeConv is used to extract features in MTYGNN.

Table VIII presents the experimental results on Paris-Lille-3D dataset. We can easily observe that DGCNN$_{\text{GAConv}}$ and MTYGNN outperform the variants without GAConv, i.e., DGCNN and MTYGNN$_{\text{EdgeConv}}$. It effectively validates that GAConv can effectively capture the hidden topology in the point cloud. In particular, GAConv can dynamically capture the structural information through the attention mechanism for better node representations.

3) Effects of GAPool: In this subsection, we explore the effects of the proposed GAPool on the performance of MTYGNN. Some variants are explored: (1) MTYGNN$_{\text{RP}}$ indicates removing all GAPool layers in MTYGNN. (2) MTYGNN$_{\text{gPool}}$ denotes applying gPool instead of the proposed GAPool.

Table IX shows the comparison results on the ShapeNet dataset. We can observe that MTYGNN and MTYGNN$_{\text{gPool}}$ are better than MTYGNN$_{\text{RP}}$, which shows the effectiveness of developing pooling methods for point clouds. Furthermore,

TABLE VI
EXPERIMENTAL RESULTS FOR EACH CLASS ON SHAPENET BENCHMARK

Methods	areo	bag	cap	car	chair	ear phone	guitar	knife	lamp	laptop	motor	mug	pistol	rocket	skate board	table	mIou
PointNet [7]	83.4	78.7	82.5	74.9	89.6	73.0	**91.5**	85.9	80.8	95.3	65.2	93.0	81.2	57.9	72.8	80.6	83.7
PointNet++ [8]	82.4	79.0	87.7	77.3	90.8	71.8	91.0	85.9	**83.7**	95.3	**71.6**	94.1	81.3	58.7	76.4	**82.6**	85.1
RGCNN [12]	80.2	82.8	**92.6**	75.3	89.2	73.7	91.3	**88.4**	83.3	96.0	63.9	95.7	60.9	44.6	72.9	80.4	84.3
DGCNN [13]	84.0	**83.4**	86.7	77.8	90.6	74.7	91.2	87.5	82.8	95.7	66.3	94.9	81.1	**63.5**	74.5	**82.6**	85.2
LDGCNN [14]	84.0	83.0	84.9	78.4	90.6	74.4	91.0	88.1	83.4	95.8	67.4	94.9	82.3	59.2	76.0	**81.9**	85.1
MTYGNN$_{\text{single}}$	**84.5**	80.3	84.9	78.8	89.9	75.2	91.2	88.2	83.4	95.7	67.8	**94.9**	81.7	62.2	**76.6**	82.2	85.4
MTYGNN	84.1	82.9	84.9	**79.4**	**91.2**	**76.8**	90.8	87.6	83.1	**96.5**	69.5	94.3	**83.7**	62.5	76.4	**82.8**	**85.7**

TABLE VII
EFFECTS OF THE Y-SHAPED MODEL ARCHITECTURE ON PARIS-LILLE-3D AND SHAPENET DATASETS

Methods	Paris-Lille-3D		ShapeNet	
	mIoU	mAcc	mIoU	OA
MTYGNN$_{\text{Single}}$	72.3	-	85.4	-
MTYGNN$_{\text{MH}}$	71.9	94.4	84.9	98.0
MTYGNN	**75.7**	**95.1**	**85.7**	**99.6**

TABLE VIII
EFFECTS OF GACONV ON SHAPENET DATASET

Baseline	mIoU	OA
DGCNN	85.1	-
DGCNN$_{\text{GAConv}}$	85.4	-
MTYGNN$_{\text{EdgeConv}}$	85.5	98.9
MTYGNN	**85.7**	**99.6**

TABLE IX
EFFECT OF GAPOOL ON SHAPENET DATASET

Baseline	mIoU	OA
MTYGNN$_{\text{RP}}$	84.9	96.0
MTYGNN$_{\text{gPool}}$	85.1	98.8
MTYGNN	**85.7**	**99.6**

the performance of MTYGNN. Moreover, the performance of MTYGNN is superior to MTYGNN$_{\text{gPool}}$. It indicates that GAPool can effectively summarize better representations by considering both structural information and feature representations during pooling.

4) Effects of Weighted Loss Function: We design several variants to evaluate the effect of loss functions with weights on model performance: (1) MTYGNN$_{\text{Add}}$ indicates that the losses of two tasks are added directly. (2) MTYGNN$_{\text{Fixed}}$ denotes using fixed weights instead of automatic learning. Here we set the weight of classification loss to 0.3 and the weight of segmentation loss to 0.7.

The experimental results on the ShapeNet dataset are presented in Table X. The results indicate that MTYGNN outperforms the other two variants. This indicates that applying homoskedastic uncertainty can automatically learn the appropriate weights for two tasks to ensure that there is no negative gain between the tasks.

TABLE X
EFFECT OF WEIGHTED LOSS FUNCTION ON SHAPENET DATASET

Baseline	mIoU	OA
MTYGNN$_{\text{Add}}$	84.9	96.0
MTYGNN$_{\text{Fixed}}$	85.1	98.8
MTYGNN	**85.7**	**99.6**

TABLE XI
ANALYSIS ON POOLING RATIO K ON SHAPENET DATASET

Pooling ratio	0.1	0.3	0.4	0.5	0.6	0.8
mIoU	82.6	85.1	85.3	85.7	85.6	85.7
OA	94.9	97.6	98.9	99.6	99.6	99.4

5) Analysis on Pooling Ratio K*:* We provide an ablation experiment to evaluate the effect of different pooling ratios K on the model performance. The results in Table XI show that the smaller the pooling ratio, the fewer nodes are retained in the point cloud and the accuracy is lower. This is because a smaller pooling ratio causes the down-sampling operation to lose important structural information in the point clouds, resulting in performance degradation. If an appropriate K value is selected, GAPool can retain the important points as much as possible, thereby preserving the most important information in the point cloud. As can be seen in Table XI, when the K value is 0.5, it still can obtain high accuracy.

6) Network Depth Study: Network depth is a critical hyper-parameter that directly determines the quality of feature learning. Our proposed MTYGNN is a multi-task learning framework, it is necessary to separately determine the network depths of the segmentation branch l_1 and classification branch l_2. Here we first remove the classification branch to obtain the best value of l_1, and then fix l_1 to obtain l_2. Table XII presents the performance for different number of network depths. From the results, we can observe that both segmentation and classification branches achieve the best performance for a network depth of 2. The performance of the model usually decreases when the network depth deviates from 2.

7) Time and Space Complexity: We measure the model complexity by comparing the model size and forward time

TABLE XII
INFLUENCE OF THE NETWORK DEPTHS

Network depths	1	2	3	4
l_1	84.4	85.7	85.4	85.1
l_2	97.2	99.6	99.2	98.8

TABLE XIII
COMPLEXITY COMPARISON

Baseline	Model size(MB)	Time(ms)	oACC(%)
PointNet [7]	35.8	11.2	83.7
DGCNN [13]	**6.2**	13.1	85.2
MTYGNN	8.5	**9.2**	**85.7**

of the proposed MTYGNN with other baseline models. For the fair comparison, all experiments are run on NVIDIA Tesla P100 GPUs. The statistical results are presented in Table. XIII, we can observe that the model size of MTYGNN is larger than DGCNN but smaller than PointNet. The reason is that MTYGNN is a multi-task learning framework that introduces multiple task branches. Furthermore, the forward time of MTYGNN is smaller than that of Pointnet and DGCNN since the proposed GAPool for downsampling operation can reduce the complexity of GAConv that has higher computational cost. In particular, the computational cost of the downstream GAConv is even negligible after multiple down-sampling. Therefore, our proposed MTYGNN has achieved the better trade-off between the model complexity, computational complexity, and accuracy.

V. CONCLUSION

Point clouds, as an efficient 3D representation, play an indispensable role in intelligent transportation scenarios. In this work, we propose a novel multi-task Y-shaped graph neural network (MTYGNN) to explore and analyze point cloud data. By extending the traditional U-shaped framework, MTYGNN performs simultaneously segmentation and classification tasks, and exploits the commonality between the two tasks to improve the generalization capability of the model. To learn better feature representation, we further design GAConv and GAPool for point clouds. GAConv exploits attention mechanisms to explore potential structural relationships in point clouds, while GAPool implements pooling operations for point clouds based on feature representations and structural information. Furthermore, we use homoscedastic uncertainty to automatically learn the weights between the two tasks to ensure that there is no negative interference between tasks. Experiments on common point cloud datasets in traffic scenarios demonstrate that MTYGNN outperforms existing benchmarks. For example, on Paris-Lille-3D, our model achieves an accuracy of 75.9%. In the ShapeNet dataset, mIoU reaches 85.9%. For future studies, we plan to investigate the effect of other features (e.g., normals, colors) in the point cloud on the model.

REFERENCES

[1] Y. Cui *et al.*, "Deep learning for image and point cloud fusion in autonomous driving: A review," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 2, pp. 722–739, Feb. 2022.

[2] Z. Luo, M. Attari, S. Habibi, and M. V. Mohrenschildt, "Online multiple maneuvering vehicle tracking system based on multi-model smooth variable structure filter," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 2, pp. 603–616, Feb. 2020.

[3] L. Ma, Y. Li, J. Li, W. Tan, Y. Yu, and M. A. Chapman, "Multi-scale point-wise convolutional neural networks for 3D object segmentation from LiDAR point clouds in large-scale environments," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 2, pp. 821–836, Feb. 2021.

[4] C. Chen, K. Li, C. Zhongyao, F. Piccialli, S. C. H. Hoi, and Z. Zeng, "A hybrid deep learning based framework for component defect detection of moving trains," *IEEE Trans. Intell. Transp. Syst.*, early access, Nov. 10, 2021, doi: 10.1109/TITS.2020.3034239.

[5] C. Chen *et al.*, "Hierarchical semantic graph reasoning for train component detection," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Mar. 31, 2021, doi: 10.1109/TNNLS.2021.3057792.

[6] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3D point clouds: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 12, pp. 4338–4364, Dec. 2021.

[7] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 652–660.

[8] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5099–5108.

[9] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.

[10] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–12.

[11] H. Gao and S. Ji, "Graph U-Nets," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2083–2092.

[12] G. Te, W. Hu, A. Zheng, and Z. Guo, "RGCNN: Regularized graph CNN for point cloud segmentation," in *Proc. 26th ACM Int. Conf. Multimedia*, Oct. 2018, pp. 746–754.

[13] 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 Trans. Graph.*, vol. 38, no. 5, pp. 1–12, Nov. 2019, doi: 10.1145/3326362.

[14] K. Zhang, M. Hao, J. Wang, C. W. de Silva, and C. Fu, "Linked dynamic graph CNN: Learning on point cloud via linking hierarchical features," 2019, *arXiv:1904.10014*.

[15] J. Liu, B. Ni, C. Li, J. Yang, and Q. Tian, "Dynamic points agglomeration for hierarchical point sets learning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 7546–7555.

[16] S. Vandenhende, S. Georgoulis, W. Van Gansbeke, M. Proesmans, D. Dai, and L. Van Gool, "Multi-task learning for dense prediction tasks: A survey," 2020, *arXiv:2004.13379*.

[17] Y. Yang, C. Feng, Y. Shen, and D. Tian, "FoldingNet: Point cloud auto-encoder via deep grid deformation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 206–215.

[18] O. Dovrat, I. Lang, and S. Avidan, "Learning to sample," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 2760–2769.

[19] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.* Cham, Switzerland: Springer, 2015, pp. 234–241.

[20] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[21] R. Cipolla, Y. Gal, and A. Kendall, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7482–7491.

[22] B.-S. Hua, M.-K. Tran, and S.-K. Yeung, "Pointwise convolutional neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 984–993.

[23] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "PointCNN: Convolution on χ-transformed points," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 820–830.

[24] J. Yang *et al.*, "Modeling point clouds with self-attention and Gumbel subset sampling," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3323–3332.

[25] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[26] X. Yan, C. Zheng, Z. Li, S. Wang, and S. Cui, "PointASNL: Robust point clouds processing using nonlocal neural networks with adaptive sampling," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 5589–5598.
[27] C. Holtz, O. Atan, R. Carey, and T. Jain, "Multi-task learning on graphs with node and graph level labels," in *Proc. NeurIPS Workshop Graph Represent. Learn.*, 2019.
[28] D. Buffelli and F. Vandin, "A meta-learning approach for graph representation learning in multi-task settings," 2020, *arXiv:2012.06755*.
[29] J. Lee, I. Lee, and J. Kang, "Self-attention graph pooling," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 3734–3743.
[30] T. Shen, T. Zhou, G. Long, J. Jiang, S. Pan, and C. Zhang, "DiSAN: Directional self-attention network for RNN/CNN-free language understanding," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1–10.
[31] X. Roynard, J.-E. Deschaud, and F. Goulette, "Paris-Lille-3D: A large and high-quality ground-truth urban point cloud dataset for automatic segmentation and classification," *Int. J. Robot. Res.*, vol. 37, no. 6, pp. 545–557, 2018.
[32] Z. Liang, M. Yang, L. Deng, C. Wang, and B. Wang, "Hierarchical depthwise graph convolutional neural network for 3D semantic segmentation of point clouds," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 8152–8158.
[33] J. Behley *et al.*, "SemanticKITTI: A dataset for semantic scene understanding of LiDAR sequences," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9297–9307.
[34] H. Su *et al.*, "SPLATNet: Sparse lattice networks for point cloud processing," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2530–2539.
[35] W. Wu, Z. Qi, and L. Fuxin, "PointConv: Deep convolutional networks on 3D point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9621–9630.
[36] Z. Wu *et al.*, "3D ShapeNets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1912–1920.

Xiaofeng Zou is currently pursuing the Ph.D. degree in computer science and technology with Hunan University, China. His research interests include machine learning, data mining, and deep learning.

Kenli Li (Senior Member, IEEE) received the Ph.D. degree in computer science from the Huazhong University of Science and Technology, China, in 2003. He has published more than 200 research papers in international conferences and journals. His major research interests include cloud computing, high performance computing, and parallel computing. He serves on the Editorial Board of the IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING (T-SUSC) and IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS (TII).

Yangfan Li is currently pursuing the Ph.D. degree in computer science with Hunan University, China. His research interests include parallel and distributed computing, machine learning, and deep learning.

Wei Wei (Senior Member, IEEE) received the Ph.D. degree from Xi'an Jiaotong University in 2010. He is currently an Associate Professor with the School of Computer Science and Engineering, Xi'an University of Technology, China. He has more than 100 papers published or accepted by international conferences and journals. His research interests include mobile computing, image processing, the Internet of Things, and distributed computing.

Cen Chen received the Ph.D. degree in computer science from Hunan University, China. He is currently a Scientist II at the Institute for Infocomm Research (I2R), Agency for Science, Technology and Research (A*STAR), Singapore. He has published several research articles in international conference and journals, such as IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS (TPDS), IEEE TRANSACTIONS ON COMPUTERS (TC), IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS (TITS), IEEE TRANSACTIONS ON CYBERNETICS (TCYB), *ACM Transactions on Knowledge Discovery from Data* (TKDD), and AAAI. His research interests include machine learning, deep learning, and parallel and distributed computing.