GT-MilliNoise: Graph Transformer for Point-wise Denoising of Indoor Millimeter-Wave Point Clouds

Walter Brescia[†a], Pedro Gomes[†b], Saverio Mascolo[a], Laura Toni[b], Luca De Cicco[a]

[a] *Politecnico di Bari, Bari, Italy*
[b] *University College London, London, United Kingdom*

Abstract

Millimeter-wave (mmWave) radars are gaining popularity thanks to their low cost and robustness in low-visibility conditions. However, the 3D point clouds they produce are sparser and noisier than those from LiDARs and depth cameras. These differences create challenges when applying existing methods, originally designed for dense point clouds, to mmWave data. Specifically, there is a gap in point-level precision tasks, such as full point cloud denoising for mmWave data, partly due to the lack of fully annotated datasets. In this work, we employ the MilliNoise dataset, a fully annotated indoor mmWave point clouds dataset, to advance the understanding of mmWave point clouds denoising via two main steps: (i) we carry out an experimental analysis of the most common point cloud processing approaches and show their limitations in exploring the local-to-global structures in sparse and noisy point clouds; (ii) in light of the identified limitations, we propose a graph-based transformer architecture, denoted as GT-*MilliNoise*, composed of two main blocks to effectively leverage both the temporal and geometric structures of the data: a *Temporal* block leverages the sparsity of data to learn the dynamic behaviour of the points; a *Geometric* block, uses a point-wise attention mechanism to form representative neighbourhoods for feature extraction. The experimental results obtained in the MilliNoise dataset show that our proposed GT-*MilliNoise* architecture outperforms the state-of-the-art both qualitatively and quantitatively. Specifically, it achieves 75% accuracy (5% gain compared to the state-of-the-art), and a significantly low Earth Mover's distance value of 0.193.

1. Introduction

In the rapidly evolving field of multimedia systems, the role of 3D data sources has grown significantly. Such data requires new approaches to capture, process, and utilize information. Point Cloud (PC) data produced, f.i., by 3D

[†]These authors contributed equally to this work.

Light Detection and Ranging (LiDAR) sensors and stereocameras, allows capturing the features of three-dimensional scenes. These technologies have driven significant advancements in applications encompassing autonomous navigation and robotics [1, 2], 3D modeling [3], augmented reality/virtual reality [4, 5, 6], and 6DoF video streaming [7, 8].

Despite their advantages, LiDAR and stereocamera systems are often challenged by environmental conditions, such as poor visibility in adverse weather or difficulties in detecting transparent or reflective surfaces. This limits the utilization of such media content for reliable machine applications. Moreover, these technologies may be cost-prohibitive and pose concerns regarding power consumption, thus hindering their adoption in mobile applications. On the other hand, mmWave radars are sensors operating with wavelengths in the order of millimeters. They are equipped with transmitter antennas, emitting signals, and receiver antennas, capturing their reflections. The processing of received signals enables the generation of a 3D PC that describes the environment, including information on the velocity, intensity, and orientation of the reflecting objects. Thanks to their operating frequencies, these sensors are significantly more robust in adverse environmental conditions, such as fog, dust, smoke, and rain [1, 5, 9, 10]. Despite such advantages, the PC data generated by mmWave sensors introduces some peculiar challenges. Specifically, *(i)* the presence of a large amount of noise, and *(ii)* the sparsity of points in each acquired frame. The higher level of noise is due to the wider wavelength of mmWave signal, which is highly susceptible to multi-path reflections. At the same time, the small antenna size and the low power signal of the mmWave lead to a lower signal-to-noise ratio (SNR), which further increases the noise and reduces the total number of points captured per frame. In indoor environments, a typical mmWave PC consists of only ~ 200 sparse points, as opposed to the tens of thousands found in LiDAR PC, and most of the points in mmWave PC are noise points. As a consequence, in a mmWave PC, shapes and objects are rarely understandable to the naked eye.

Given these unique challenges, it is not obvious if current PC processing approaches can be extended for the mmWave PCs. Most of the existing PCs learning architectures were developed for significantly denser and less noisy PCs generated by LiDAR sensors or RGB-D cameras. Such architectures have started to be adapted for mmWave PCs, but mainly focusing on PC level tasks such as action recognition [6, 11, 12] or broad-level object detection [13, 14]. Conversely, understanding how well current architectures from dense PCs behave with mmWave data for point level tasks (e.g., point-wise denoising) is highly overlooked and remains a gap in the literature.

This work addresses this gap and advances the research on point-wise mmWave PC denoising. To this end, we leverage MilliNoise [15], a dataset with point-wise labels with a sub-millimetric accuracy. Frequency Modulated Continuous Waves (FMCW) mmWave sensors with integrated antennas, referred to as Antenna-On-Package (AoP), are used for the data campaign of this dataset. MilliNoise collects *indoor* scenarios, such as wide and narrow hallways, tight and loose turns, and shelves built by properly arranging a set of obstacles. Most im-

portantly, each point in the MilliNoise dataset is accurately labelled through a motion capture system, allowing the discrimination of points describing actual objects in the scene from those representing noise.

Building on the MilliNoise dataset, this work aims at developing learning models for point-wise denoising of mmWave PCs. To the best of our knowledge, this represents the first attempt at point-wise denoising tailored for the unique challenges of indoor mmWave PCs. To achieve this, we first investigate the performance of several state-of-the-art approaches typically used for traditional PC processing that, however, have never been investigated for denoising highly sparse and noisy mmWave PCs. Specifically, we implement the state-of-art point-based [16, 17], graph-based [18], and transformer-based [19] architectures. The results obtained show that state-of-the-art methods exhibit limitations in exploring the local-to-global structures present in sparse PC and modelling the relations between points. To address these limitations, we further propose GT-*MilliNoise*, an architecture for mmWave PC denoising composed of two main cascading blocks:

1. A *Temporal* block, designed to explore the dynamic aspect of the input PC by learning how to distinguish between sporadic points (likely to be noise) and consistent points over time.
2. A *Geometric* block, which takes this temporal information into self-attention operation to form representative neighbourhoods able to explore the 3D space. From these neighbourhoods, a message-passing graph convolution captures relations between the points to learn spatial-temporal features for the denoising task.

The proposed GT-*MilliNoise* is able to denoise mmWave scenes with an average 0.68 F1-score and 75% accuracy, corresponding to a 5% gain over the state-of-the-art. Beyond the quantitative gain in terms of accuracy (i.e., percentage of correct points), we also observe a significant gain in geometric metrics, which measure how similar the denoised scene is to the target ground truth. Simulation results demonstrate that the GT-*MilliNoise* model better understands the 3D scene. As a last contribution, we conduct an ablation study to investigate how the input size and the inclusion of additional features (*i.e.*, velocity, intensity), as well as each component of the GT-*MilliNoise* architecture, affect the model performance. We conclude this paper by highlighting potential directions for improving research.

In summary, this work makes the following contributions:

- A novel GT-*MilliNoise* method able to process mmWave PCs for point-wise denoising. It compensates for noisy and sparse data by learning temporal features, which reflect point consistency over time, and extracting spatio-temporal features with a second *Geometric* block (Section 4). This two-stage approach allows the GT-*MilliNoise* to effectively leverage both temporal and geometric cues for accurate denoising.

- To the best of our knowledge, the first benchmarking on deep learning methods for indoor point-wise denoising (on the MilliNoise dataset),

proposing an in-depth analysis of how different architectures handle mmWave data (Section 6).

- An ablation study of both input modalities and network architecture, shedding light on how they impact the model performance (Section 6.5).

2. Background

This section describes Frequency-Modulated Continuous Waves (FMCW) radar PCs acquisition (Section 2.1) and provides a brief overview of the main classic approaches for PC processing (Section 2.2) that do not focus on mmWave data, hence avoiding its peculiar issues. Finally, it describes how these methods were adapted to mmWave PC data, specifically focusing on the denoising task (Section 2.3).

2.1. FMCW Radar data acquisition

mmWave radars emit a set of signals (*chirp frame*), and based on their reflection, build a set of two-dimensional arrays describing the intensities of the received signals. By leveraging the Time-of-Flight (ToF), the displacement of the receiving antennas, and the Doppler effect, it is possible to extract a 3D PC representing the position of the objects, the intensity of its reflected signal and the radial relative velocity [20]. The interested reader is referred to [20] for an extensive overview of mmWave FMCW radar processing.

FMCW mmWave radars are gaining popularity for PC acquisition due to their superior ability to penetrate 3D space in low visibility conditions and their low price, making them easily accessible. They also adopted integrated antennas, resulting in significantly reduced production and retail costs, which helped increase the adoption of mmWave sensors in several research and industrial fields. However, the constraints on the signal and on the sensor's antennas limit the quality of the acquired PC representation. The mmWave wider wavelength is highly susceptible to multi-path interference and scattering. The interference effect is exacerbated in confined indoor spaces where multi-path reflections are formed more easily.

Outdoor PCs vs Indoor PCs: with respect to dense and outdoor PCs, the first difference stems from the distance between the objects reflecting the antenna's signals. In fact, given the higher distance between objects (from at most a few meters in the indoor scenario to tens of meters in outdoor cases), the effect of reflections from other objects is dramatically reduced. Furthermore, in the outdoor case, the higher distance between objects prevents (or at the very least reduces) the chance of interfering reflections, increasing the overall quality of the point cloud. By contrast, in indoor scenarios, objects are much closer, and the chance of signals interfering with each other is much higher, resulting in a much more noisier PC. This is further proved by the percentage of noise points in the MilliNoise dataset (see Section 3). A second fundamental difference relies on point-wise labelling: outdoor datasets often lack a point-wise accurate labelling methodology. The lack of accurate point-wise labels prevents the computation

of any kind of loss function or metrics for both carrying out the neural network training and post-training evaluation.

2.2. Related works on Point Clouds Processing

PCs are a set of unordered and irregular points. Processing such PCs to extract key information is challenging due to the lack of regular structures, which makes the usage of deep neural networks more involved. A common methodology to address this issue is to develop networks to handle raw PCs, without pre-processing steps (e.g., voxelization) that could obscure natural invariances of the data or introduce quantization errors. This approach was pioneered by PointNet [16] and PointNet++ [17] architectures, which proposed to process each point independently and aggregate the output via an invariant function. However, this point-based strategy fails to consider relationships between points when learning features. To address this issue, Dynamic Graph Convolution (DGCNN) [18] proposes to process the PC as a graph and to use a Graph Neural Network (GNN) to extract features by processing the edges between points. More recently, transformer-based architecture gained popularity for PC processing. The transformer considers the input PC as a fully connected graph. By considering all points, the Transformer can learn by itself the neighbourhood of each point for feature extraction. Transformer architectures, while powerful, may struggle to capture complex relationships between points in comparison with GNN, which are specifically designed to learn features from edges between points. In [21], authors propose a multi-head self-attention mechanism based on learnable projections and rasterization and de-rasterization operations, composing a "cloud transform block". The proposed architecture is evaluated on a dense PC for semantic segmentation, object classification, PC inpainting and image-based reconstruction. However, these tasks do not match the scope of this work, which is the point-wise classification of sparse indoor PCs. In [22], authors present a point transformer block, which is used in combination with k-nearest neighbours (knn) layers and multilayer perceptrons. Later, in [23], the knn layer is replaced with a serialized neighbor mapping. Compared to our work, both architectures were evaluated on significantly denser PCs (from hundreds of thousands to millions of points per frame with respect to few hundreds points of a mmWave frame) for semantic segmentation and object classification tasks. In the recent years, several approaches have emerged to tackle the task of point cloud denoising [24, 25, 26, 27]. In [24], authors propose an approach based on Reinforcement Learning to train multiple neural network modules which are chosen dynamically to best denoise the input point clouds. The approach is based on the identification of local *patches*, which are defined by parameters set empirically, to choose which neural network module to leverage for denoising. However, the identification of *patches*, i.e. local structures, is strongly impaired on mmWave PCs due to the points' sparsity, in contrast to dense point clouds derived from meshes. Authors of [25] leverage a structure based on PointNet++ to denoise the point cloud, integrated with features coming from DINOv2, a diffusion model, that extracts information from RGB cameras. However, in our work, the main objective is to denoise point clouds coming from FMCW

radars, without the use of support sensors, rendering [25] unsuitable. The work presented in [26] introduces a joint normal filtering approach which leverages a neural network to apply *shape-aware* noise removal. However, both the concepts at the base of this approach, i.e. normal filtering and shape-awareness, are inappropriate for mmWave point clouds due to the sparseness and noisiness of data. In [27], authors propose a DGCNN based on *invertible neural networks* to apply *multilayer graph convolution*. Also this approach strives to identify local structures in the input (dense) point clouds, which are, however, hardly present in sparse mmWave PCs. The approaches described above were originally designed for dense and relatively noise-free PCs from LiDAR sensors or RGB-D cameras. However, the ability of these models to extract key features from PCs for downstream tasks does not directly translate to mmWave sensors, due to the higher levels of noise and sparsity in the mmWave PCs. The Transformer architecture, while capable of capturing long-range dependencies between points, does not model these relationships and their complexity in the same way as message-passing GNN. Since it learns keys, queries and values ~~V~~ independently of the point neighbourhood, the self-attention operation is not as expressive as other operations, such as graph message-passing convolution, that incorporate graph-edge information, f.i., by learning a message between two points. Another work which tackles mmWave PCs elaboration is presented in [28]. Authors propose a methodology based on diffusion, assisted with a second modality (a LiDAR) to carry out super resolution of mmWave PCs. The *denoising* process applied in this work is done with an auto-encoder (an U-net) whose aim is to reproduce a LiDAR-like PC, therefore removing possible noise points. However, the denoising process is not tackled directly. Moreover, authors tackle outdoor scenarios. Another work addressing mmWave PCs super resolution is presented in [29]. The work leverages raw ADC data to carry out (after two accumulating steps) PC elaboration. However, extracting such data from the sensor requires an additional module[1], while our work aims at using the mmWave radar out-of-the-box. Secondly, authors leverage a LiDAR to carry out the mmWave PC labelling, considering the mmWave points close to a LiDAR points as true points, false otherwise. In our work, the ground truth PCs are labelled with a sub-millimetric accuracy, without relying on another modality (see Section 3). In the next subsection, we describe the approaches applied specifically to mmWave PC processing, with a clear focus on the denoising task.

2.3. Related Works on mmWave Point Clouds Denoising

In current literature, mmWave PCs processing has been approached from two directions: 1) for human sensing applications and 2) for autonomous navigation applications. In the former, the developed methods process datasets [6, 14, 11, 13, 12] consisting of sequences of human activities acquired using a *static* mmWave radar, with the final task of action recognition. By contrast, autonomous navigation methods, including ours, process data acquired from

[1] `https://www.ti.com/tool/MMWAVEICBOOST`

mmWave radar mounted on a *moving system*, such as a robot or vehicle. As the agent moves, the surrounding 3D space is captured as PC, which is then processed to facilitate navigation tasks [30].

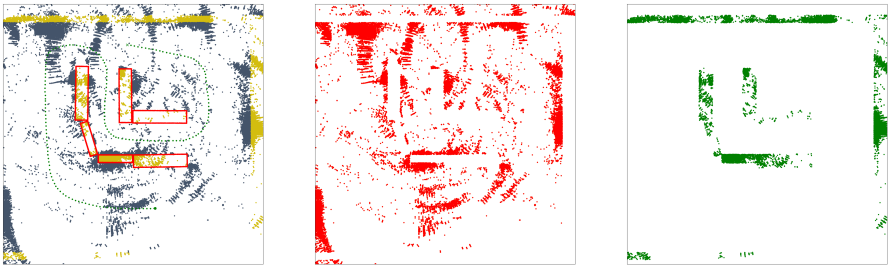
Both applications share a common challenge of space-time feature extraction in highly sparse and noisy 3D environments, typical of mmWave acquisition systems. However, the type of noise and sparsity and the related impact on each task are different. Given the moving sensor and the environments with multiple objects, navigation PCs typically have more false/noise points compared to PCs for action recognition. This makes denoising for navigation more challenging, as it requires accurate labelling of each individual point. By contrast, action recognition only needs a single label for the whole point cloud sequence.

Despite this challenge, in the current literature, there is a lack of datasets of mmWave with accurate labels for denoising tasks. The lack of datasets is due to the difficulty of creating high-fidelity ground-truth annotations. Given this issue, the annotation process is usually simplified. The common approaches use a second modality as a reference (e.g. LiDAR) [31, 28], are based on acquisition characteristics (e.g. Doppler effect) [32], or manually label selected points of interest [33]. However, these labelling strategies are usually incomplete and lack precision, with no guarantee of being an accurate ground truth (or label). Due to this lack of annotated data, most methods are measured based on some broad PC-level (or subset) detection task instead of accurate point-level metrics. In [34], a convolutional neural network (CNN) and PointNet++ are implemented for mmWave PC denoising of outdoor scenes. However, the authors only consider the points extremely close to the sensor. In [2, 35] a Similarity Group Proposal Network (SGPN) [36] and a PointNet++ [17] were used, respectively. However, only a selection of points was labelled in the dataset [33] and used in these works. A large number of points were simply labelled as "background" and nenglected. A complete denoising of the mmWave PCs was done in [31]. To make this possible, the authors create an algorithm to annotate the mmWave PCs dataset using a secondary LiDAR sensor. Given the fully annotated data, they implemented a simple PointNet architecture, achieving a 0.73 F1-score. However, all the above works process PCs acquired from outdoor scenes. To the best of our knowledge, our work is the first to explore *indoor navigation* PC denoising.

As seen in Section 2.2, most proposed architectures for mmWave PCs denoising are simple PointNet and PointNet++. These are usually modified, f.i. replacing the sampling step with mean shift clustering [41] or adding more processing blocks [42, 31]. However, the backbone of PointNet point-based operations remains constant. Looking more broadly at processing PC networks, there are several works [10, 9, 1, 43] outperforming point-based architectures with more advanced architectures such as graph-based or transformer-based deep neural networks. However, those works do not focus on point-wise denoising tasks. They are also usually considered in multi-modal settings, processing a combination of mmWave data with 2D RGB images or LiDAR. Within this context, the MilliNoise dataset provides the opportunity to benchmark existing methods and develop new ones for point-wise denoising tasks. As shown in Ta-

Dataset	Runs	Frames (Points)	Environment	Annotation
nuScenes [30]	1000	1.3M (N/A)	Outdoor	Bounding Box
Astyx [37]	N/A	500 (N/A)	Outdoor	Object oriented
CARRADA [38]	N/A	12.6k (35M)	Outdoor	Bounding Box
OdomBeyondVision [39]	119	N/A (N/A)	Indoor	Not annotated
RadarScenes [40]	158	N/A (N/A)	Outdoor	Point-Wise; Labels for 10 Objects class; 'Others' class
Radar-Ghost [2]	111	N/A (35M)	Outdoor	Point-Wise; Labels for 5 Objects class; 'Background' class
MilliNoise [15]	49	58k (11.6M)	Indoor	Point-Wise sub-mm accuracy; Labels for True or False class; distance-to-object label

Table 1: Comparison of mmWave Datasets for autonomous navigation tasks



(a) Complete scene with obstacle boundaries (red), the real points (yellow), and noise points (blue).

(b) Noise Points

(c) True Points

Figure 1: Example of scenario of the MilliNoise dataset (Scenario number 3).

ble 1, MilliNoise is the first mmWave PC dataset of *labelled indoor* scenes with a sub-millimeter accuracy. In this paper, we benchmark the most popular point-based architectures, identify key limitations, and propose a novel architecture, namely GT-*MilliNoise*, outperforming existing ones in the denoising task.

3. The MilliNoise Dataset

This section briefly describes the MilliNoise dataset[2], initially introduced in [15], that will be used to benchmark the proposed architectures and the ones from the literature.

The details of the dataset are summarized in Appendix A.

The MilliNoise dataset collects 49 individual runs captured in 6 different real-world scenarios totalling around 58k frames for a total duration of 8 hours

[2] `https://github.com/c3lab/MilliNoise`

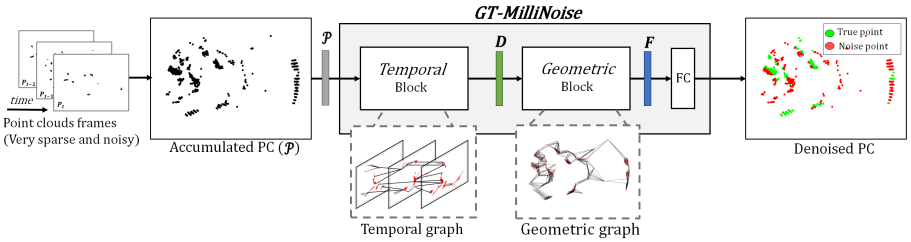


Figure 2: Proposed pipeline architecture for PC denoising composed of two main blocks: *Temporal* and *Geometric* and example how on the PC is converted to graph in each block

of data collection. This allows to carry out in-depth exploration of temporal dynamics in the point clouds and its effect on noise points. In total, around 12 million individually labelled 3D points have been collected, with 60.15% of noise points (7 million of points) and the remaining part of clean points (5 million). Besides its (x_i, y_i, z_i) coordinates, each point is equipped with velocity and intensity information, which paves the way to analyses investigating the impact of that additional information on the noise points.

4. mmWave Point Cloud Denoising

This section proposes the GT-*MilliNoise* approach for mmWave PC denoising. We start by defining the denoising problem and then present the GT-*MilliNoise* method, discussing first its strategy to learn features from mmWave PCs and later its architecture details.

Problem Formulation:

We tackle the denoising problem as a binary classification task. Given an input PC frame $P_t = \{p_1, \cdots, p_M\}$ defined as a set of points $p_i = \{x, y, z\} \in \mathbf{R}^3$, the model outputs the probability of the point representing a true/real obstacle or false/noise obstacle. This score is then used to calculate the *Binary Cross Entropy* (BCE) used as training loss. In this work we include in the definition of *false* points, i.e. points representing fake obstacles, all those points that do not overlap with the ground truth, including the points generated from ghost targets, clutter, and propagation phenomena. While those points may still carry information about the environment from a signal processing point of view, they are not useful for robot navigation, since they would still represent obstacles in the wrong position.

4.1. Proposed GT-MilliNoise Approach

Our goal is to extract features of mmWave PCs, which is made challenging by the sparsity and the high amount of noise points in the scene. Given these challenges, the GT-*MilliNoise* strategy is to:

1. increase the input data to the model accumulating a number of consequent frames (*frame accumulation*);

2. exploit the sparsity of each PC frame to learn the point's temporal behaviour (*temporal processing*);
3. employ a point-wise attention mechanism to build representative neighbourhoods for feature extraction and to model the large amount of noisy and sporadic points in the 3D scene (*geometrical processing*).

The GT-*MilliNoise* approach starts by taking an accumulating window of PC frames as the model input. Formally, we define the *Accumulated Point Cloud* $\mathcal{P}(t, W)$, at a time-stamp t, as the PC obtained by accumulating a window of W sequential PC frames together as follows: $\mathcal{P}(t, W) = (P_{t-W+1}; \cdots; P_t) \in \mathbf{R}^{N \times 3}$ with $N = WM$ points, where each frame is associated with a timestamp from the vector $T = \{t - W + 1, \ldots, t\} \in \mathbf{R}^{W \times 1}$. Intuitively, accumulating PCs from multiple time steps not only increases the amount of geometric information available but also provides a temporal description of the scene. To leverage both these temporal and geometric features, the augmented PC is processed by two main processing blocks: (1) a *Temporal* block and (2) a *Geometric* block.

The goal of the *Temporal* block is to extract the temporal information of the scene from the input accumulated PC. However, extracting such information from PC data is challenging since there is no explicit point-to-point correspondence over time. This challenge is further exacerbated in the case of mmWave PCs due to their high sparsity. Nonetheless, we argue that the same sparseness of data can be leveraged to extract valuable temporal information. In fact, we can exploit the sparsity to capture the notion of *point temporal consistency*. Specifically, a point is considered "*consistent*" over time (in W) when its 3D position is regularly occupied at other time steps. In contrast, "*sporadic*" points appear abruptly in empty areas and vanish almost instantly. Figure 3 presents an example of a sporadic (in yellow) and a consistent (in blue) point in time. We know mmWave PCs have a significantly higher amount of sporadic points compared to traditional point clouds. Our intuition is that while consistent points could be associated to actual objects, the sporadic points are more likely due to noise and are generally not critical for understanding the scene's geometry. The *Temporal* block is a set of network layers designed to extract this temporal aspect and is presented in Section 4.2.1.

Concerning the *Geometric* block, its goal is to capture the geometric structures within the PC data. In PC literature, it has been shown that such structures can be captured by aggregating information from geometric neighbourhoods. However, such a process is not as straightforward for mmWave PCs. The common assumptions for traditional PC do not necessarily apply to mmWave PCs. For instance, since in mmWave PCs false points can still fall close to true points, the ones in close proximity in the geometric space do not necessarily belong to the same latent object space. This makes it very challenging to define effective neighbourhoods and, within those neighbourhoods, to aggregate information, given that a significant percentage of the points are false and unreliable. To address these issues, the *Geometric* block employs a self-attention mechanism to dynamically find informative neighbourhoods from each point. For this operation, the attention mechanism leverages the temporal features learned in

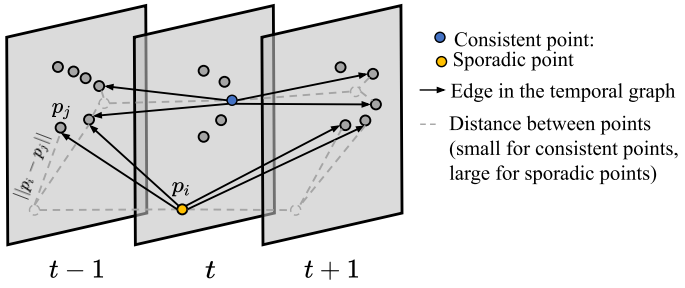


Figure 3: Example of how the edges are built in the temporal graph $\mathcal{G}_\mathcal{T}$ processed by the *Temporal* block. The point in blue (yellow) is an example of a consistent (sporadic) point in time.

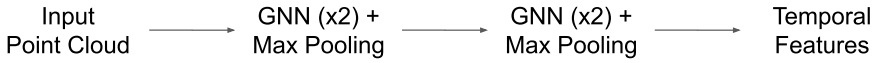


Figure 4: *Temporal* block architecture

the *Temporal* block, which identifies the sporadic and unreliable points for understanding the scene's geometry. Using the learned attention, the *Geometric* block constructs a graph representation, which allows us to extract features from informative neighbourhoods and model complex relationships between points connected by edges. The *Geometric* block is explained in detail in Section 4.2.2.

4.2. GT-MilliNoise Architecture

In the following, we describe the proposed GT-*MilliNoise* architecture for PC denoising, depicted in Figure 2. GT-*MilliNoise* takes in input the accumulated point cloud $\mathcal{P}$. Then, a first *Temporal* block builds a graph from the accumulated point cloud frames, from which it learns dynamic features $D = \{d_1, ..., d_N\} \in \mathbf{R}^{N \times C_D}$. The learned dynamic features are then sent along with the input PC to a *Geometric* block that builds a *geometric* graph to learn geometric features $F = \{f_1, ..., f_N\} \in \mathbf{R}^{N \times C_F}$. In the last step, the learned geometric features are concatenated to learn a global representation and are processed by multiple fully connected (FC) layers, which outputs a classification score for whether a point is a real point or a noise point. For the detailed architecture description, we refer the reader to Appendix C.

4.2.1. Temporal Block

At its core, the *Temporal* block is a message-passing GNN [44]. However, unlike conventional GNNs that aggregate features from spatial neighbourhoods, the *Temporal* block aggregates features from temporal neighbourhoods. To this end, the *Temporal* block starts by building a temporal graph $\mathcal{G}_\mathcal{T} = (\mathcal{V}, \mathcal{E})$ by considering each point in $\mathcal{P}$ as a node in the graph and building edges between points across time. Specifically, for each point p_i of the P_{t_α} frame at time step t_α, we build an edge to its k geometrically closest neighbours in the other frames

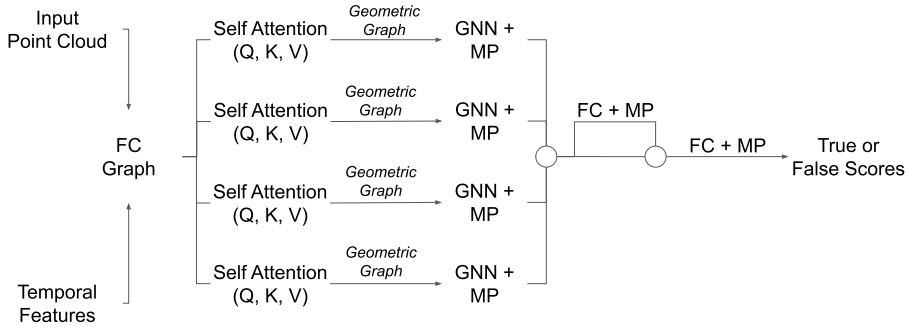


Figure 5: Schematic of the *Geometric* block. (FC: Fully Connected; MP: Max Pooling)

(P_{t_β} with $t_\alpha \neq t_\beta$). An example of temporal graph $\mathcal{G}_\mathcal{T}$ is depicted in Figure 3, while a simplified version of the architecture is presented in Figure 4.

After the temporal graph is built, the *Temporal* block performs a message-passing convolution to aggregate information from the temporal neighbourhoods. More specifically, for the α-th frame, for each i-th point of the l-th layer, the *Temporal* block concatenates: *(i)* the i-th point temporal feature d_i^{l-1}; *(ii)* the temporal feature d_j^{l-1} from its j-th neighbourhood, from time-step t_β, *(iii)* the geometric distance $\|p_i - p_j\|$ and *(iv)* the time displacement $\|t_\alpha - t_\beta\|$, between point i and j. This large concatenation is processed by a learnable layer $\Theta_\mathcal{T}^l$ to learn the message m_{ij}^l, modelling the relation between the two points.

$$m_{ij}^l = \Theta_\mathcal{T}^l\big(d_i^{l-1}; d_j^{l-1}; \|p_i - p_j\|; \|t_\alpha - t_\beta\|\big) \tag{1}$$

Next, for each point i, its neighborhood messages m_{ij}^l are aggregated via max pooling $\bigoplus$ to compute the temporal feature $d_i^l \in D^l$ as follows.

$$d_i^l = \bigoplus_{j \in \mathcal{N}_{\mathcal{T}\,i}} \{m_{ij}^l\} \tag{2}$$

After the *Temporal* module extracts the temporal features D, we combine those features with the original stacked PC $\mathcal{P}$ and feed them to the *Geometric* block as shown in Figure 2. Given the inclusion of temporal features, the *Geometric* block performs a neighbourhood aggregation with the prior notion of each point's temporal consistency. In Section 6.5.3, we investigate the individual impact of the *Temporal* block on the overall GT-*MilliNoise* performance.

4.2.2. Geometric Block

The *Geometric* block takes the accumulated PC $\mathcal{P}$ and temporal features D as input and outputs geometric features F for each point, as depicted in Figure 5. Each layer l of the *Geometric* block is a combination of a self-attention operation and a message-passing operation. In each layer, we first learn the neighbourhoods using the self-attention mechanism. Then, using the learned

attention, we build a *geometric* graph $\mathcal{G}_c$, from where a message-passing operation extracts geometric features. In the following we explain each operation in detail.

Self-Attention operation: The goal of the self-attention operation is to learn how each point relates to another. Thus, the self-attention mechanism begins by learning *queries* ($Q \in \mathbf{R}^{N \times C_Q}$), *keys* ($K \in \mathbf{R}^{N \times C_K}$, with $C_Q = C_K$), and *values* ($V \in \mathbf{R}^{N \times C_V}$), as the linear projection of the concatenation of the input accumulated PC. An *attention matrix* ($A \in \mathbf{R}^{N \times N}$) is computed by taking the dot product of the query vectors and the key vectors and then dividing by the square root of the dimension of the query vectors. The end result is an attention matrix containing a scalar value representing the *point attention* with respect to all the other points.

Lastly, we learn the refined values $\hat{V} \in \mathbf{R}^{N \times C_V}$ by performing a weighted sum of value vectors, where the weights are determined by the attention matrix.

$$\hat{V} = AV, \quad A = \text{softmax}\left(\frac{QK^T}{\sqrt{C_K}}\right) \tag{3}$$

This weighted aggregation is the core of the Transformer architecture [19]. However, since it learns values V independently of the point neighbourhood, this operation is not as expressive as graph message-passing convolution, which incorporates edge information by learning a message between two points. As such, to model complex relations between points, a graph message-passing convolution is employed after the self-attention mechanism.

To this end, a *geometric* graph $\mathcal{G}_c$ is built based on the learned attention values. Specifically, for each point i, we build an edge connecting it to the top k points with the highest attention matrix A_i to point i. In contrast to the temporal graph, where edges only connect points across a single time step, the geometric graph has learned attention values for every point-to-point relationship in the PC. This grants the model the flexibility to dynamically define informative neighbourhoods for feature aggregation among all possible combinations. Furthermore, for each point, this attention-selected neighbourhood varies at each layer of the *Geometric* block, enabling the network to explore the 3D space at varying scales (from local to global) to capture the geometric structure within the data.

Message-passing operation: Given the geometric graph, the network performs a message-passing convolution by processing the concatenation of: *(i)* the values v_i^l, *(ii)* the refined values $\hat{v}_i^l$, *(iii)* the values related to the neighbors v_j^l and *(iv)* the learned attention scalar between points A_{ij}.

$$m_{i,j}^l = \Theta_m^l\left(v_i^l; \hat{v}_i^l; v_j^l; A_{ij}^l\right) \tag{4}$$

The geometric features (f_i^l) for each point are then computed as the message

K-Fold:	Training Scenarios	Test Scenarios
Fold-1-6	*All* [1-6]	*All* [1-6] (19 runs)
Fold-123	[4,5,6]	[1,2,3] (10 runs)
Fold-4	[1,2,3,5,6]	4 (10 runs)
Fold-5	[1,2,3,4,6]	5 (6 runs)

Table 2: The considered train and test splits.

aggregation.

$$f_i^l = \bigoplus_{j \in \mathcal{N}_i} \{m_{ij}^l\} \tag{5}$$

By combining the self-attention and message-passing operations, at each layer $l-th$ the *Geometric* block takes advantage of the best of both approaches. The self-attention enables the model to explore the 3D neighbourhoods to capture local-to-global geometric structures (later shown in Section 6.4), and the message-passing enables the model to learn complex relations between points.

5. Implementation

This section describes how the model is implemented and the experimental conditions.

5.1. Training

We tackle the denoising problem as a binary classification task. For every point $P_i \in \mathcal{P}$, the model learns the probability $p_\theta(Y_i)$ of a point being a true point ($Y_i = 1$) or being a false point ($Y_i = 0$). Therefore, the proposed GT-*MilliNoise* and benchmarking methods are trained using the common *Binary Cross Entropy* (BCE) as a loss function, as follows.

$$BCE = -\frac{1}{N}\sum_{i=0}^{N} Y_i \log(p_\theta(Y_i)) + (1 - Y_i)\log(1 - p_\theta(Y_i)) \tag{6}$$

where Y_i is the ground-truth label for the point i.

K-Fold Train/Test Splits: To provide a robust evaluation of our proposed method, we employ a cross-validation approach to split the dataset into multiple train/validation/test sets. We then train the same model on each of the considered data splits and average their performance. The splits are presented in Table 2. As anticipated in Section 3, the MilliNoise dataset consists of 6 scenarios, with each scenario being one unique obstacles course navigated by the robot. Then, for each scenario, different runs have been considered, generating multiple and diverse robot paths per scenario. The splits in Table 2 have been created in order to evaluate the model performance across different scenarios. For example, in the split named *Fold-4*, the model is trained on runs from scenarios $1, 2, 3, 5, 6$ and evaluated on runs from scenario 4. The exception

to this rule is the split named *Fold*-1-6. In this split, the same scenarios are shared for the train/validation/test sets. It is worth mentioning that even when one scenario is shared across training, validation and testing, different runs are used for different sets, preventing any data leakage. This is because the PCs are captured from the robot's reference frame, and each individual run follows a particular trajectory within the scenario, making each run unique. As a consequence, different runs across train/validation/test sets in the *Fold*-1-6 imply that the model is evaluated on previously unseen data.

5.2. Evaluation Metrics

To evaluate the performance of our models on the denoising task, we consider BCE, accuracy, recall, and F1-score on the test dataset. While the above metrics are widely used in classification tasks, they are not sufficient to measure the quality of the denoised PC. Specifically, these metrics are averaged across all points. This means that regions with a high density of points have a significantly higher impact on the final metric. However, for tasks such as robot navigation, *not all points have the same importance.* While it is still desirable to have a high percentage of correctly classified points, it is more critical to correctly classify the points that actually describe obstacles in order to avoid them. Therefore, we consider additional *geometric metrics*, in order to compare the geometry of the growth-truth PC with the estimated one.

Geometric metrics for denoising quality: Given a complete run of the robot across one scenario, we define P^{true} as the set of points in PC that are true objects, and $\hat{P}^{true}$ as the point classified by the model as real by the model, as follows.

$$P^{true} = \{p_i \in P_t \quad | \quad label(p_i) = \text{true point}\} \tag{7}$$

Hence, P^{true} is the true representation of the scenario, and $\hat{P}^{true}$ is what the model perceives as the scenario. Our goal is to measure the geometric difference between these two PCs. The smaller the differences, the better the model's understanding of the scene. To this end, we use both the Chamfer distance (CD) [45] and Earth's Moving Distance (EMD) [46]. These metrics are widely used in the literature for PC comparison and are defined as follows.

Chamfer distance (CD): The CD measures the distance between each point in the predicted PC and its closest target point in the reference PC, and vice-versa.

$$CD(P^{true}, \hat{P}^{true}) = \frac{1}{n} \sum_{p_i \in P} \min_{\hat{p_i} \in \hat{P}} ||p_i - \hat{p}_i||^2 + \frac{1}{n} \sum_{p \in \hat{P}} \min_{p_i \in P} ||\hat{p}_i - p_i||^2 \tag{8}$$

Earth's moving distance (EMD): The EMD between two distributions (or PCs in this case) is proportional to the minimum cost required to change one distribution into the other. More formally, the EMD solves an optimization problem by finding the optimal point-wise bijection mapping θ between two PCs, namely

$\theta : P^{true} \rightarrow \hat{P}^{true}$. The EMD distance is then given by the distance of the points at both ends of this mapping, as follows:

$$EMD(P^{true}, \hat{P}^{true}) = \min_{\theta: P^{true} \rightarrow \hat{P}^{true}} \sum_{p_i \in P^{true}} ||p_i - \theta(p_i)||^2. \tag{9}$$

Notice that the EMD requires the PCs in input to share the same size. Since the number of true points in the ground truth PC and the true points identified by the model can be different, we downsampled both the complete PC runs to $1,000$ points using farthest-point-sampling (FPS). We found this sampling value to provide a good overall representation of the PC geometry, simplifying the optimization task without affecting the final score.

It is worth noting that the EMD and CD were only used during the evaluation phase and not during training since their implementation as loss function is not trivial. In particular, these metrics are only applied to a subset of the input PC (P^{true}, the true points). As such, these metrics are very susceptible to the number of true points in the input PC. However, in PCs from the MilliNoise, the number of true points varies greatly from frame to frame, with cases lacking true points, causing the geometric loss values to be very volatile (high variation) during training.

5.3. Bechmarking Methods

One of the contributions of our work is the benchmarking of classic PC processing methods designed for traditional PCs, on mmWave PCs from the MilliNoise dataset. For this analysis, we selected architectures representing the main PC processing strategies, which are point-based, graph-based and attention-based and represent the common base for most of the *dense* point cloud denoising techniques.

PointNet [16]: PointNet is able to learn directly from PC data by processing each point independently via a shared learnable function and aggregating the output.

PointNet++ [17]: extends PointNet point-based operation to hierarchical neighbourhoods by sampling the PC between layers. PointNet and PointNet++ are implemented following the original papers.

Dynamic Graph Convolutional Network (DGCNN)[18]: DGCNN is a GNN that learns features by processing the PC as a graph. The DGCNN is implemented following the original paper [18].

Transformer [19]: While the first Transformer architecture was designed for text-data processing, its ability to learn unstructured data quickly lead to it's adaptation for PC processing by multiple works. For the sake of generality, during benchmarking, we implemented a "*vanilla*" Transformer architecture. Our benchmarking Transformer is composed of four self-attention layers in parallel as described in the *Geometric* block (Section 4.2.2).

5.4. GT-MilliNoise Architecture Details

For all the considered architectures, a batch size of 32 is selected, the point neighbourhood is set to 8 points ($k = 8$), and hyper-parameters are fine-tuned

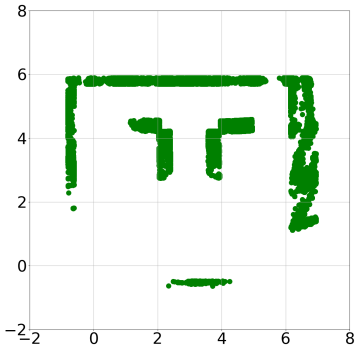
for each architecture. Batch normalization is applied at the end of each layer. The models are trained with a learning rate of either 0.01 or 0.001 using ADAM optimizer, depending on architecture and accumulation window W. The GT-*MilliNoise* is implemented with four layers, each layer learning features with 64 dimensions. For more details on the implementation, we refer the reader to the Appendix C. The code is made publicly available[3].

6. Results

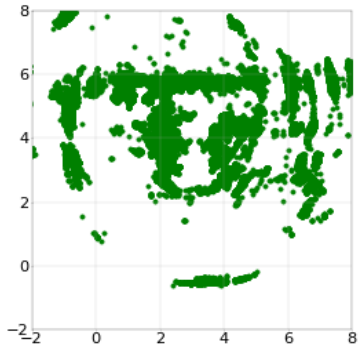
This section presents and analyzes the results obtained from the architectures on denoising the MilliNoise dataset. Before carrying out an in-depth analysis, in Section 6.1 we first highlight how classical classification metrics and geometrical metrics can lead to conflicting results. Next, in Sections 6.2 and 6.3, we provide a comprehensive analysis to shed more light on the advantages and drawbacks of the considered architectures when dealing with mmWave PCs. Section 6.4 investigates how each considered architecture explores the 3D space to extract features. Section 6.5 presents an ablation study to analyze the impact of the different aspects of the input data as well as the aspects of the model architecture on performance. Finally, Section 6.7 presents the limitations of this work and discusses possible future research directions.

6.1. Understating of Accuracy and Geometric Metrics

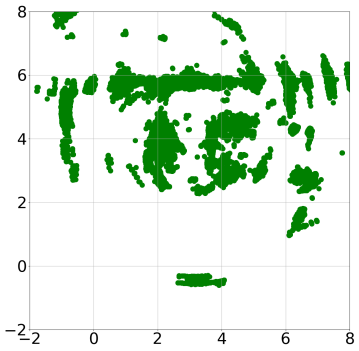
In the Figures 6 and 7, we show the visual denoising results of the different models for two runs of the robot. The points are roto-translated to a global reference frame to facilitate understanding, and the caption reports the accuracy and EMD value for each model. At a quick glance, it can be seen that examples with the best (highest) accuracy do not necessarily have the best (lowest) EMD values. An example of this discrepancy can be seen in Figure 6, between PointNet and PointNet++, and in Figure 7, between PointNet and DGCNN. In the former instance, the PointNet (Figure 6.a) was able to denoise a run in scenario 1 with 66% accuracy. In comparison, PointNet++ (Figure 6.b) has 61%, and the Transformer (Figure 6.e) has 67% accuracy. However, despite the lower accuracy, the PointNet++ and Transformer visualizations are more similar to the ground truth scene than PointNet visualization. This "geometric similarity" is instead better captured by the EMD metric, where PointNet++ and Transformer have significantly lower EMD compared to PointNet. This mismatch between metrics occurs because the accuracy measures how many individual points are correct and, as such, is biased by high-density regions. On the other hand, the EMD measures how similar the predicted PC is to the ground truth and, hence, how well the model understood the overall scene.



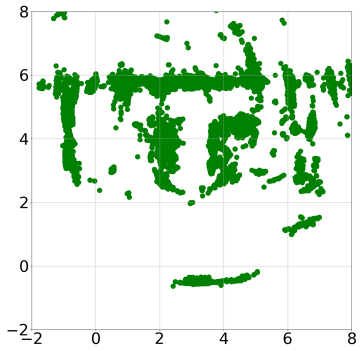
(a) Ground Truth - Real Points



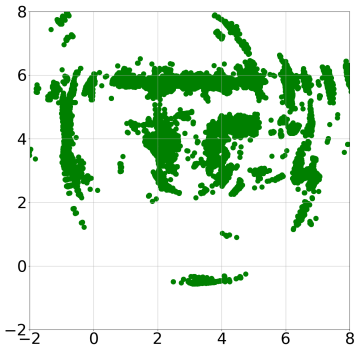
(b) PointNet (Acc: 66% , EMD: 0.65)



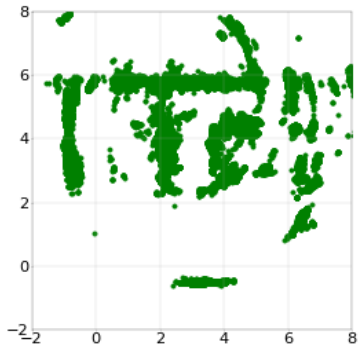
(c) PointNet ++ (Acc: 61% , EMD: 0.52)



(d) DGCNN (Acc: 68% , EMD: 0.50)

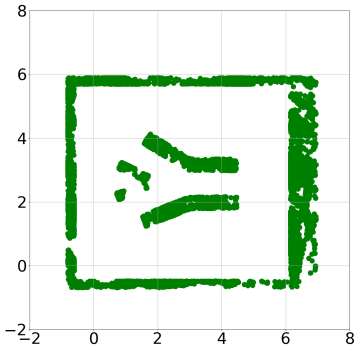


(e) Transformer (Acc: 67%, EMD: 0.53)

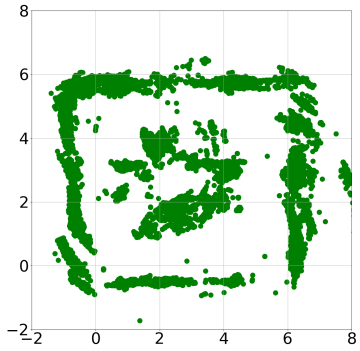


(f) GT-*MilliNoise* (Acc: 69% , EMD: 0.48)

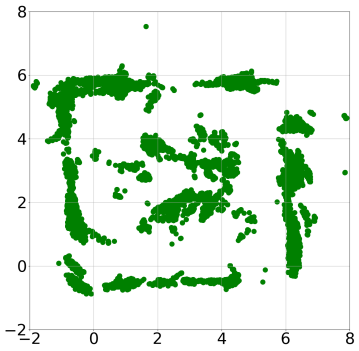
Figure 6: Denoising results of Scene 1 (*Fold-123* run 4).



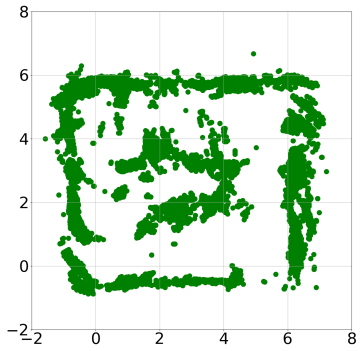
(a) Ground Truth - Real Points



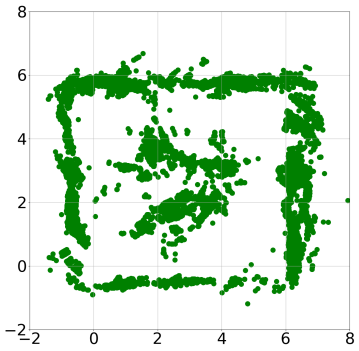
(b) PointNet (Acc: 76% , EMD: 0.61)



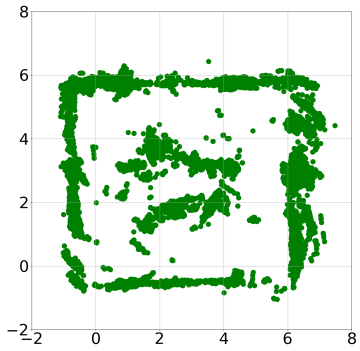
(c) PointNet++ (Acc: 72% , EMD: 0.50)



(d) DGCNN (Acc: 75% , EMD: 0.44)



(e) Transformer (Acc: 78% , EMD: 0.41)



(f) GT-*MilliNoise* (Acc: 81%, EMD: 0.33)

Figure 7: Denoising results of Scene 4 (*Fold-4* run 70).

Architecture	BCE ↓	Acc. ↑	Recall ↑	F1 ↑	EMD ↓	CD ↓
PointNet¸	0.661	0.710	0.585	0.613	0.434	0.221
PointNet++	0.640	0.667	0.464	0.520	0.499	0.243
DGCNN	0.656	0.708	0.600	0.618	0.427	0.214
Transformer	0.578	0.700	0.604	0.613	0.418	0.210
GT-*MilliNoise* (ours)	**0.550**	**0.750**	**0.674**	**0.681**	**0.360**	**0.193**

Table 3: Performance results averaged across folds

6.2. Overall Results

The visual results from Figures 6 and 7 clearly show the proposed GT-*MilliNoise* is better at denoising the scene compared to benchmarking methods. While the state-of-the-art approaches can capture the overall representation of the scenario (walls and objects), the GT-*MilliNoise* are clearly closer to the ground truth and are visually cleaner. Specifically, the object's boundaries are "sharper," and the paths where the robot can move safely are mostly unobstructed.

The GT-*MilliNoise* superior performance is validated by the quantitative results obtained. Table 3 presents average results across train/test splits, given an accumulating frame window length of $W = 12$. The results show that the proposed GT-*MilliNoise* architecture consistently outperforms all the remaining architectures, achieving 75% accuracy and an F1-Score of 0.68. In comparison, the state-of-the-art architectures PointNet, DGCNN, and Transformer achieved an average$\sim$ 70% accuracy and F1-score of ~ 0.61. PointNet++ was an exception, achieving an underwhelming 64% accuracy. Considering the difficulty of the task, the 70% accuracy achieved by most state-of-the-art architectures is a positive result. These performance show these architectures originally designed for traditional PCs can be applied to indoor mmWave PCs for early results.

However, despite the success of the state-of-art approaches, the GT-*MilliNoise* still represents a substantial improvement over them. Although the GT-*MilliNoise* accuracy gain might seem modest ($\sim$ 5%), the GT-*MilliNoise* has an overall better understanding of the 3D scene. This better understanding is represented by the significantly lower EMD of 0.36, in comparison with the benchmarking methods EMD of $\sim$ 0.43. This better understanding of the scene translates visually to cleaner object shapes with less noise and distortion, as can be seen in the denoising scenes from Figures 6 and 7. Additionally, the GT-*MilliNoise* lower BCE value indicates that even misclassified points are associated with lower confidence, enabling the recognition of uncertain regions while maintaining high accuracy for the ones with high confidence.

6.3. Generalization to unseen scenarios - K-Folds Results

To understand how well the model can generalize to unseen scenarios, we investigate the performance for the different train/test splits in Tables 4 and 5.

[3] `https://github.com/LASP-UCL/GT-MilliNoise`

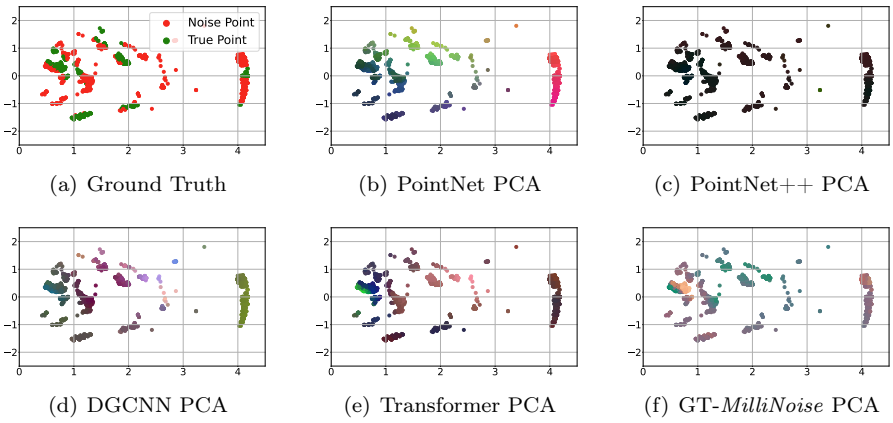
Evaluation of Scenes: 1,2,3,4,5,6 (Fold-1–6)						
Architecture	BCE ↓	Acc. ↑	Recall ↑	F1 ↑	EMD ↓	CD ↓
PointNet	0.536	0.740	0.623	0.651	0.387	0.222
PointNet++	0.619	0.659	0.350	0.445	0.580	0.255
DGCNN	0.562	0.712	0.625	0.628	0.444	0.226
Transformer	0.499	0.750	**0.725**	0.693	0.405	0.209
GT-*MilliNoise* (ours)	**0.480**	**0.794**	0.695	**0.725**	**0.309**	**0.179**
Evaluation of Scene: 4 (Fold-4)						
Architecture	BCE ↓	Acc. ↑	Recall ↑	F1 ↑	EMD ↓	CD ↓
PointNet	0.659	0.734	0.629	0.629	0.490	0.214
PointNet++	0.592	0.693	0.505	0.542	0.478	0.247
DGCNN	0.652	0.725	0.620	0.617	0.447	0.214
Transformer	0.546	0.733	0.584	0.611	0.450	0.214
GT-*MilliNoise* (ours)	**0.542**	**0.761**	**0.644**	**0.660**	**0.443**	**0.198**

Table 4: Performance results on K-fold data split for Scenes 1–6 (Fold-1–6) and Scene 4 (Fold-4).

Evaluation of Scenes: 1,2,3 (Fold-123)						
Architecture	BCE ↓	Acc. ↑	Recall ↑	F1 ↑	EMD ↓	CD ↓
PointNet	0.712	0.694	0.562	0.592	0.452	0.237
PointNet++	0.691	0.650	0.541	0.550	0.536	0.265
DGCNN	0.697	0.713	0.572	0.612	0.387	0.213
Transformer	0.635	0.654	0.537	0.551	0.377	0.209
GT-*MilliNoise* (ours)	**0.593**	**0.725**	**0.695**	**0.666**	**0.328**	**0.197**
Evaluation of Scene: 5 (Fold-5)						
Architecture	BCE ↓	Acc. ↑	Recall ↑	F1 ↑	EMD ↓	CD ↓
PointNet	0.736	0.670	0.525	0.580	0.406	0.211
PointNet++	0.657	0.664	0.460	0.543	0.402	0.206
DGCNN	0.713	0.683	0.584	0.615	0.430	0.202
Transformer	0.646	0.662	0.576	0.596	0.439	0.208
GT-*MilliNoise* (ours)	**0.583**	**0.720**	**0.662**	**0.672**	**0.361**	**0.196**

Table 5: Performance results on K-fold data split for Scenes 1–3 (Fold-123) and Scene 5 (Fold-5).

The *Fold-1-6* results are relevant in applications where the robot navigates in "familiar" scenarios, such as a factory robot trained on its specific floors. Under these train/test conditions, the GT-*MilliNoise* achieves a high accuracy equal to $\sim 80\%$. To note, the test scenarios are different trajectories than the training one, ensuring the model is evaluated on unseen data. The results from the *Fold-123; Fold-4; Fold-5* are relevant for applications where the robot operates in scenarios completely different from its training data. Even in this more challenging setting, our model achieved positive results with denoising accuracy ranging from 72% to 76%. These results demonstrate the GT-*MilliNoise* ability to generalize to unseen scenarios, indicating its potential for broader real-world



(a) Ground Truth (b) PointNet PCA (c) PointNet++ PCA

(d) DGCNN PCA (e) Transformer PCA (f) GT-*MilliNoise* PCA

Figure 8: Features PCA of each models (PCs with $W = 12$ and plotted from the robot perspective).

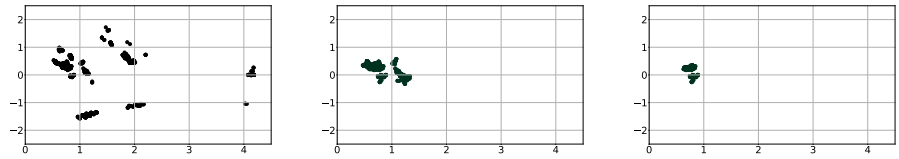
applications.

6.4. 3D Space Exploration Discussion

To better understand the obtained results and how traditional PC approaches handle mmWave data, this section investigates how each architecture explores the 3D space to extract features. From this investigation, we highlight each architecture's main benefits and shortcomings when dealing with mmWave PCs.

To this end, for each model, we project the high-dimensional features from the last layer of the neural netwok – before being converted into classification scores – into a low-dimensional domain using Principal Component Analysis (PCA). In particular, the last layer has been reduced to three variables (through PCA) which are mapped, after min-max normalization, to RGB colour. Given a PC as input, Figure 8 shows for each architecture the PCA of the learned features as point colour. Figure 9 shows the points classified as true by each model. Figure 10 shows a zoom-in of the PCA in a particular region of interest. In the figures, similar colours represent points with similar high-dimensional features. The ground-truth point cloud depicted in Figure 8 (a) shows that the true and noise points are grouped in separate small communities/structures. Hence, a good representation should have very distinct values (i.e. colours) between clusters of true and noise points. By contrast, a smooth (or completely absent in the worst case) colour gradient reflects a model with poor performance that failed to identify local communities/structures in the PC. Using this understanding and the PCA visualizations from Figure 8, we now explain the inner workings of each architecture and its advantages and limitations.

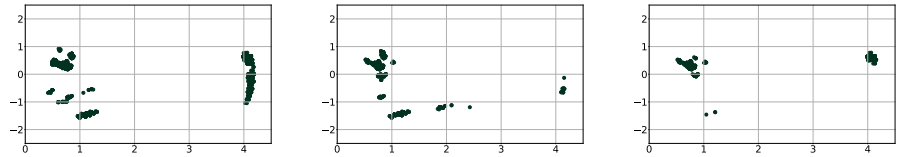
PointNet: Figure 8 (b) shows that the features learned by the PointNet are smooth across space. This is expected since each point feature is learned from the point's 3D coordinates, neglecting the point's neighbourhood. Given its



(a) Ground Truth (true points)

(b) PointNet prediction

(c) PointNet++ Prediction

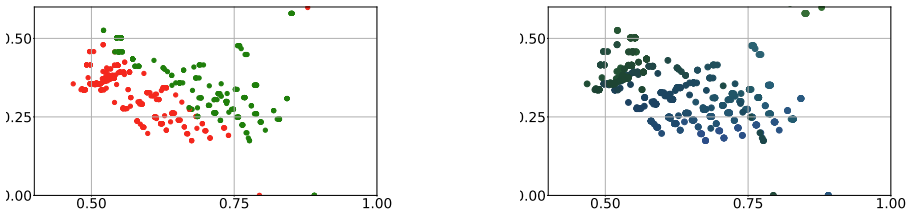


(d) DGCNN Prediction

(e) Transformer Prediction

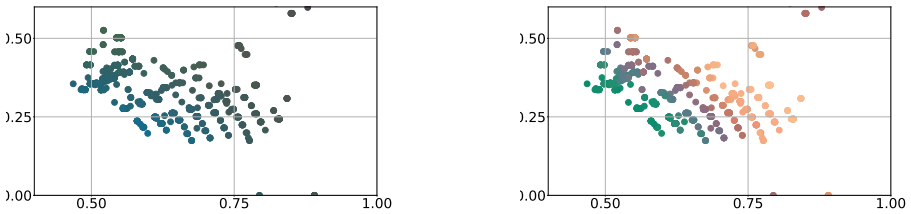
(f) GT-*MilliNoise* Prediction

Figure 9: Prediction of each model (PCs with $W = 12$ and plotted from the robot perspective).



(a) Ground-Truth PC

(b) PointNet PCA (Smooth color)



(c) DGCNN PCA (Smooth color)

(d) GT-*MilliNoise* PCA (Sharp color)

Figure 10: Zoom-in visualization in a region of interest from the PCA examples in Figure 8

inability to capture the local structures, the PointNet identifies correctly one main region/cluster. The nearby points are smoothly classified as true, while the far points are smoothly classified as false. In most cases, the selected cluster is close to the sensor position ($x = 0, y = 0$), as in the prediction in Figure 9 (b). This cluster has a higher probability of being true and is high-density (given the sensor's physical properties). As such, its correct classification results in high accuracy but is not necessarily associated with a good global scene understanding.

PointNet++: This architecture, widely used in classical PCs, was designed to address PointNet's inability to capture local-to-global structures through hierarchical learning. The hierarchical learning should be possible via the sampling of the PC in initials layers and the consequent interpolation in the later layers. However, this has a side effect of over-smoothing, which is too extreme for the particular case of MilliNoise dataset. This is observed first in Figure 8 (c), in which features from PointNet++ are almost all identical, and then in the prediction (Figure 9 (c)), where the model misses the proper classification of many true points.

DGCNN: As shown in Figure 8 (d), the DGCNN is able to detect local clusters within the data. However, the features of the points within these clusters are over-smoothed. The zoom-in in Figure 10 (c) depicts a cluster with over-smoothed features. This over-smoothing is a result of the DGCNN strategy to rebuild the neighbourhood at each layer using feature similarity. This strategy, combined with the natural smoothing aspect of GNNs, causes the neighbourhoods of each point to remain fixed at each layer. Given a fixed neighbourhood, a point in the DGCNN only explores its local cluster to learn features. As such, the DGCNN is unable to efficiently explore local-to-global geometrical structures within the mmWave PC, leading to a poor performance.

Transformer & ***Graph-Transformer***: In Figure 8 (e) and (f), we observe the PCA of features from the Transformer and the GT-*MilliNoise* (which combines attention and message-passing). Both have sharp colour variations between points. The sharp variation means that the model is able to learn highly distinct features between points of different classes (noise and true points) in close geometric proximity, an ability the models investigated above do not possess. This ability is achieved by learning attention maps to explore the 3D space. The learned attention grants the network flexibility to explore local-to-global structures for each point. The ability to distinguish local structures is especially noticeable in the cluster close to the sensor, depicted in the zoom-in in Figure 10. There is a very clear correspondence between the ground-true visualization and GT-*MilliNoise* PCA features, demonstrating GT-*MilliNoise* is able to distinguish between true and false point structures.

The superior results of GT-*MilliNoise* are a direct result of this attention-based strategy to explore the space, which is well-suited for sparse and noisy mmWave PCs. This strategy, combined with message-passing convolution, allows for more effective exploration and understanding of the mmWave dataset's

Input	Architecture	BCE ↓	Acc ↑	Recall ↑	F1 ↑
Input: 1 frame	PointNet	0.629	0.693	0.599	0.602
	PointNet++	0.680	0.613	0.382	0.434
	DGCNN	**0.591**	0.688	0.512	0.560
	Transformer	0.623	0.666	0.464	0.519
	GT-*MilliNoise* \o-*Temp*	0.633	**0.716**	**0.651**	**0.640**
Input: 3 frames	PointNet	0.604	0.720	**0.657**	0.646
	PointNet++	0.626	0.654	0.300	0.403
	DGCNN	0.602	0.679	0.459	0.526
	Transformer	0.535	0.730	0.590	0.629
	GT-*MilliNoise*	**0.520**	**0.768**	0.584	**0.662**
Input: 6 frames	PointNet	0.634	0.691	0.504	0.560
	PointNet++	0.687	0.755	0.598	0.655
	DGCNN	0.567	0.716	0.621	0.630
	Transformer	0.525	0.745	0.680	0.675
	GT-*MilliNoise*	**0.502**	**0.782**	**0.711**	**0.717**
Input: 12 frames	PointNet	0.536	0.740	0.623	0.651
	PointNet++	0.619	0.659	0.350	0.445
	DGCNN	0.562	0.712	0.625	0.628
	Transformer	0.499	0.750	**0.725**	0.693
	GT-*MilliNoise*	**0.480**	**0.794**	0.695	**0.725**

Table 6: Results obtained varying the number W of stacked PC frames (for K-fold-1-6).

spatial structure.

6.5. Ablation Studies

We now turn our investigation to how the input data and the Temporal and Geometric blocks affect the model's denoising performance. In particular, we perform an ablation study on the *(i)* number of frames in the accumulation window; *(ii)* the inclusion of velocity and intensity in input PC; *(iii)* the impact of the *Temporal* block; and (*iv)* the different *Geometric* block implementations.

6.5.1. The effect of accumulated point cloud frames

In this subsection, we investigate the effect of PC sparseness on the performance of the considered architectures. Specifically, we test the denoising task with different W values, which represent the number of point cloud frames accumulated in the input point cloud $\mathcal{P}$. Table 6 presents the results for $W = \{1, 3, 6, 12\}$, with higher W leading to denser input PCs. The results show that increasing W generally leads to a denoising accuracy improvement, which is expected since increasing W increases the amount of geometrical structures available to the models.

An exception is the PointNet++ model. PointNet++ fails to denoise PCs with $1, 3, 12$ frames, however for $W = 6$ the model achieves an accuracy of 75% and an F1-score of 0.63. In this instance, a point cloud of 6 frames, comprising

Input	Architecture	BCE ↓	Acc ↑	Recall ↑	F1 ↑
xyz	PointNet	0.536	0.740	0.623	0.651
	PointNet++	0.619	0.659	0.350	0.445
	DGCNN	0.562	0.712	0.625	0.628
	Transformer	0.499	0.750	**0.725**	0.693
	GT-*MilliNoise*	**0.480**	**0.794**	0.695	**0.725**
xyz + intensity	PointNet	0.588	0.693	0.578	0.594
	PointNet++	0.610	0.678	0.444	0.517
	DGCNN	0.646	0.622	0.167	0.256
	Transformer	0.563	0.703	0.464	0.548
	GT-*MilliNoise*	**0.450**	**0.796**	**0.726**	**0.734**
xyz + velocity	PointNet	0.564	0.711	0.584	0.610
	PointNet++	0.605	0.678	0.525	0.558
	DGCNN	0.513	0.757	0.671	0.681
	Transformer	0.489	0.759	0.670	0.683
	GT-*MilliNoise*	**0.448**	**0.804**	**0.695**	**0.734**

Table 7: The effect of additional variables on the model performance (for K-Fold-1-6) .

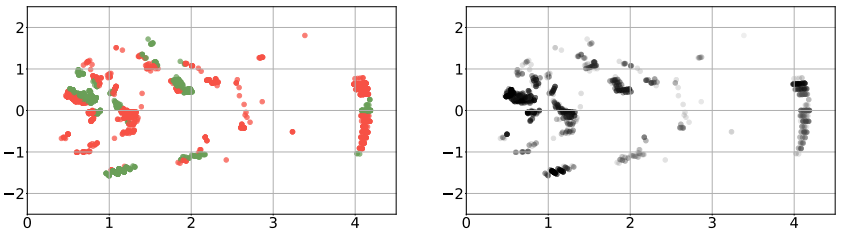
$1,200$ points, shares the same number of points of the original PointNet++ paper configuration. These results show how fragile and configuration-dependent the PointNet++ is in denoising mmWave PCs.

6.5.2. The effect of including velocity and intensity

We now investigate the effect of various features available from the radar sensor in the denoising task. Table 7 reports the model performance when given the additional intensity or velocity in input.

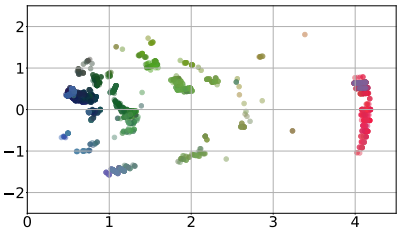
A small performance improvement is registered by including information on the point velocity in the model input. For example, GT-*MilliNoise* improves accuracy from 79% to 80% and F1-score from 0.72 to 0.73. The intuition is that, due to the nature of the signal noise, real points will share similar velocity values. On the other hand, noise points will have very different values with respect to their neighbour points since they are mainly produced by multi-path reflections. This creates PCs with regions of points with similar (very diverse) values for true (noise) points. To note, the GT-*MilliNoise* architecture was not designed to explore the velocity. The velocity is included in the model input via simple concatenation with the point's coordinates. It is possible that a model specifically designed to exploit velocity could achieve even better performance.

Table 7 also shows, contrary to expectations, that the inclusion of intensity as input to the model does not result in improved performance. The reason for the low-performance gain is that, due to the main source of noise being multiple wave reflections, noise points would share a low-intensity value. However, in an indoor environment, this is not the case: the indoor nature of the environment reduces the distance travelled by the waves, resulting in small to no reduction of the intensity of the waves received from noise points and true points. Hence,

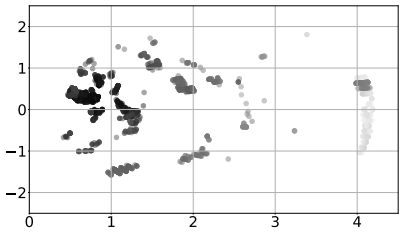


(a) Ground-truth PC

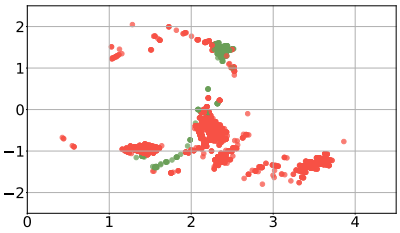
(b) Pseudo temporal consistency



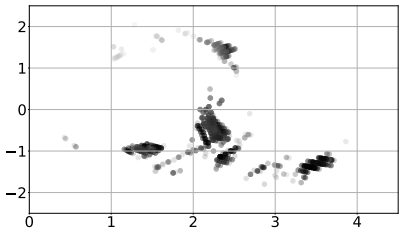
(c) PCA of dynamic features to 3 dimensions



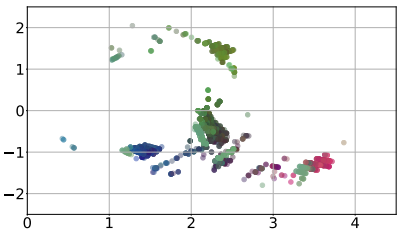
(d) PCA of dynamic features to 1 dimension



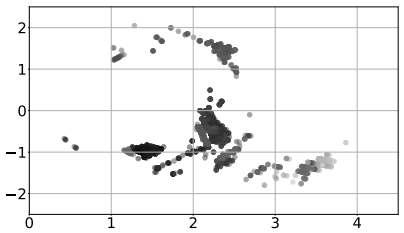
(e) Ground-truth PC



(f) Pseudo temporal consistency



(g) PCA of dynamic features to 3 dimensions



(h) PCA of dynamic features to 1 dimension

Figure 11: Comparison of pseudo-temporal consistency (darker colour represents a stronger temporal consistency) with the features learned on *Temporal* block projected to 3 and 1 dimensions. Figure from two PCs examples with $\mathcal{W} = 12$ acquired from the robot reference system.

GT-*MilliNoise*	BCE ↓	Acc. ↑	Recall ↑	F1 ↑	EMD ↓	CD ↓
without $Temp$	0.575	0.727	0.660	0.657	0.409	0.208
with $Temp$	**0.535**	**0.749**	**0.667**	**0.685**	**0.360**	**0.192**

Table 8: The improvement gained by the Temporal-block (average across the considered train/test folds)

Architecture	BCE ↓	Acc. ↑	Recall ↑	F1 ↑	EMD ↓	CD ↓
KHop-GNN	0.563	0.739	0.629	0.656	0.403	**0.198**
GT-*MilliNoise*	**0.535**	**0.749**	**0.667**	**0.685**	**0.360**	0.192

Table 9: Comparison of the *Graph-Transformer* and a *KHop-GNN* (average across the considered train/test folds).

the addition of such information to the input PC does not improve performance.

6.5.3. The Impact of Temporal Block

In this subsection, we investigate the effect of the *Temporal* block in the denoising task. We begin by evaluating a variation of GT-*MilliNoise* without the *Temporal block*, denoted as "GT w\o-$Temp$". The comparison results are reported in Table 8 and empirically show that the inclusion of the temporal block leads to significant improvement across all metrics.

To gain a deeper insight into the *Temporal* block, in Figure 11, we visualize the dynamic features learned by the block and compare them to a *pseudo-temporal consistency* computed with the ground truth by averaging, for each point, the distance of the closest points in the immediate past and future frames. Figure 11 depicts two different point clouds. In particular, the Figure shows from left to right: the ground-true PC; *pseudo-temporal* consistency computed; and the PCA features learned from the *Temporal* block, projected to 3 and 1 dimensions respectively. In Figures 11 (b) and (f), a darker colour corresponds to a higher consistency of the point in time, while a lighter colour means the point is sporadic. A visual correspondence can be found comparing the calculated pseudo-temporal consistency and PCA features. The darker points in Figure 11 (b) from the pseudo-temporal consistency correspond to darker blue points in Figure 11 (c) and (g) (3 dimensions PCA) and to darker points Figure 11 (d) and (h) (1 dimension PCA). This visual correspondence shows the *Temporal* block is learning how consistent is each point in time, producing more robust and informative features. These features are then passed to the *Geometric* block, which utilizes it to learn attention weights that take into account if points are stable or not.

6.5.4. ***Geometric Block Design Choice:***

In Table 9, we experimented with a different architecture design for the *Geometric* block. We investigated if, instead of using learned attention, which can be computationally expensive, we can build neighbourhoods in a *ad-hoc* manner. To this end, we implemented an architecture denoted as *KHop-GNN*.

The *KHop-GNN* is a message-passing GNN, where the neighbourhoods increase by a *K-hop* at each layer. For example, at the first layer, the network learns point features to its 1^{st} closest neighbourhoods, whereas at the $l-th$ layer, the network aggregates from the k^{th} closest neighbourhoods. More details about *Khop-GNN* architecture can be found in Appendix D.

Table 9 shows it was possible to approximate the performance of the GT-*MilliNoise* with a *KHop-GNN*, which achieved $\sim 74\%$ accuracy. However, it is important to note that the design *Khop-GNN* has to be manually tuned. The neighbourhood (the jump between layers) has to be manually pre-defined and is dependent on the size of the input point cloud, whereas the GT-*MilliNoise* can handle PC of any size.

6.6. Computational Complexity

We evaluated the computational complexity of the proposed GT-*MilliNoise* model and found it comparable to state-of-the-art architectures. The GT-*MilliNoise* contains 1.6M trainable parameters, which in the context of PC processing is considered a moderate-size model. It can process the entire training dataset in 243 seconds on single Tesla V100S-PCIE GPU with 32 GB of dedicated memory. These results mean the GT-*MilliNoise* model can be considered for practical real-world applications. A detailed comparative analysis of the computational complexity of the different models and components is reported in Appendix B, Table B.10.

6.7. Limitations and Future Work

We believe our method can be improved in several ways. Firstly, the denoising performance of the GT-*MilliNoise* can be further improved. While it is a good early approach, with a significant improvement over the state-of-the-art, Figures 6, 7, and 9 show our proposed model still misclassified a considerable amount of points. Secondly, the metrics to evaluate the performance of the denoising models have a limited ability to measure how well the scene is understood. We have partly addressed this challenge by proposing a metric that evaluates the geometry of the denoised point cloud (EMD and CD) instead of per-point accuracy. However, such metrics are still point averages, with every point having the same impact on the final score, and more representative metrics can be designed. Moreover, we did not take advantage of the distance to the closest object associated with each point provided by MilliNoise dataset. We believe this metric can be incorporated into the loss function to train more sophisticated and robust models. Another possible direction that leverages such information might be to cast the problem from a binary classification into a regression or a clustering-based one, in order to give space to some tolerances.
Lastly, we highlight that mmWave radars PCs are quite sensitive to the sensor's parameter configuration and that the MilliNoise dataset only includes data coming from one configuration. Although this limitation is inherent for any learning based approach, we note that such a bias may only hinder performance of the specific trained model under different sensor configurations (or mmWave sensor

altogether), but not the methodology as a whole. We leave the above directions as future works.

7. Conclusion

This paper is focused on point-wise denoising of mmWave PCs, where the goal is to classify every point either as true or false. To this end, we introduced MilliNoise [15], a dataset with fully annotated indoor mmWave PCs. We then benchmarked the performance of the most common point cloud processing approaches on this dataset and showed their limitations in exploring the local-to-global structures in sparse and noisy point clouds. To address the identified limitations, we proposed a graph-based transformer architecture denoted as GT-*MilliNoise* for mmWave denoising. The proposed GT-*MilliNoise* is a two-stage architecture able to leverage both temporal and geometric cues for accurate denoising, achieving 75% accuracy, corresponding to 5% gain over the state-of-the-art. The superior performance of GT-MilliNoise was confirmed in visual results, where the denoised PCs appeared clearer with sharper object boundaries. Additionally, geometric metrics such as EMD demonstrated that GT-MilliNoise has a superior understanding of the 3D space.

8. Acknowledgements

This work has been funded by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART", CUP: D93C22000910001).

References

[1] H. Yin, X. Xu, Y. Wang, R. Xiong, Radar-to-lidar: Heterogeneous place recognition via joint learning, Frontiers in Robotics and AI 8 (2021). doi:10.3389/frobt.2021.661199.

[2] F. Kraus, N. Scheiner, W. Ritter, K. Dietmayer, The Radar Ghost Dataset – An Evaluation of Ghost Objects in Automotive Radar Data, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021, pp. 8570–8577. doi:10.1109/IROS51168.2021.9636338.

[3] V. Sanchez, A. Zakhor, Planar 3d modeling of building interiors from point cloud data, in: Proceedings of the 19th IEEE International Conference on Image Processing, 2012, pp. 1777–1780. doi:10.1109/ICIP.2012.6467225.

[4] G. Zhang, X. Geng, Y.-J. Lin, Comprehensive mpoint: A method for 3d point cloud generation of human bodies utilizing fmcw mimo mm-wave radar, Sensors 21 (19) (2021). doi:10.3390/s21196455.
URL `https://www.mdpi.com/1424-8220/21/19/6455`

[5] Y. Almalioglu, M. Turan, C. X. Lu, N. Trigoni, A. Markham, Milli-rio: Ego-motion estimation with low-cost millimetre-wave radar, IEEE Sensors Journal 21 (3) (2021) 3314–3323. doi:10.1109/JSEN.2020.3023243.

[6] S. Palipana, D. Salami, L. A. Leiva, S. Sigg, Pantomime: Mid-air gesture recognition with sparse millimeter-wave radar point clouds, in: Proceedings of the ACM Interact. Mob. Wearable Ubiquitous Technol., Vol. 5, 2021, pp. 1–27. doi:10.1145/3448110.
URL https://doi.org/10.1145/3448110

[7] B. Han, Y. Liu, F. Qian, Vivo: visibility-aware mobile volumetric video streaming, in: Proceedings of the 26th ACM Annual International Conference on Mobile Computing and Networking, MobiCom '20, 2020, pp. 1–13. doi:10.1145/3372224.3380888.
URL https://doi.org/10.1145/3372224.3380888

[8] I. Viola, P. Cesar, Chapter 15 - volumetric video streaming: Current approaches and implementations, in: G. Valenzise, M. Alain, E. Zerman, C. Ozcinar (Eds.), Immersive Video Technologies, Academic Press, 2023, pp. 425–443. doi:https://doi.org/10.1016/B978-0-32-391755-1.00021-3.

[9] C. X. Lu, S. Rosa, P. Zhao, B. Wang, C. Chen, J. A. Stankovic, N. Trigoni, A. Markham, See through smoke: robust indoor mapping with low-cost mmwave radar, in: Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services, MobiSys '20, 2020, p. 14–27. doi:10.1145/3386901.3388945.
URL https://doi.org/10.1145/3386901.3388945

[10] Y. Cheng, H. Xu, Y. Liu, Robust small object detection on the water surface through fusion of camera and millimeter wave radar, in: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2021, pp. 15243–15252. doi:10.1109/ICCV48922.2021.01498.

[11] A. Sengupta, F. Jin, R. Zhang, S. Cao, mm-Pose: Real-Time Human Skeletal Posture Estimation Using mmWave Radars and CNNs, IEEE Sensors Journal 20 (17) (2020) 10032–10044. doi:10.1109/JSEN.2020.2991741.

[12] A. D. Singh, S. S. Sandha, L. Garcia, M. Srivastava, Radhar: Human activity recognition from point clouds generated through a millimeter-wave radar, in: Proceedings of the 3rd ACM Workshop on Millimeter-Wave Networks and Sensing Systems, mmNets'19, 2019, p. 51–56. doi:10.1145/3349624.3356768.
URL https://doi.org/10.1145/3349624.3356768

[13] H. Cui, S. Zhong, J. Wu, Z. Shen, N. Dahnoun, Y. Zhao, MiliPoint: A Point Cloud Dataset for mmWave Radar (2023). arXiv:2309.13425.

[14] Y.-H. Wu, H.-C. Chiang, S. Shirmohammadi, C.-H. Hsu, A Dataset of Food Intake Activities Using Sensors with Heterogeneous Privacy Sensitivity Levels, in: Proceedings of the 14th Conference on ACM Multimedia Systems, MMSys '23, 2023, p. 416–422. doi:10.1145/3587819.3592553.

[15] W. Brescia, P. Gomes, L. Toni, S. Mascolo, L. De Cicco, Millinoise: a millimeter-wave radar sparse point cloud dataset in indoor scenarios, in: Proceedings of the 15th ACM Multimedia Systems Conference, MMSys '24, Association for Computing Machinery, 2024, p. 422–428. doi:10.1145/3625468.3652189.
URL https://doi.org/10.1145/3625468.3652189

[16] C. R. Qi, H. Su, K. Mo, L. J. Guibas, 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.

[17] C. R. Qi, L. Yi, H. Su, L. J. Guibas, Pointnet++: Deep hierarchical feature learning on point sets in a metric space, in: Proceedings of the Conference on Neural Information Processing Systems, 2017, p. 5105–5114.

[18] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, J. M. Solomon, Dynamic graph cnn for learning on point clouds, ACM Transactions on Graphics 38 (5) (oct 2019). doi:10.1145/3326362.
URL https://doi.org/10.1145/3326362

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 30, Curran Associates, Inc., 2017, p. 51–56.

[20] X. Li, X. Wang, Q. Yang, S. Fu, Signal processing for tdm mimo fmcw millimeter-wave radar sensors, IEEE Access 9 (2021) 167959–167971.

[21] K. Mazur, V. Lempitsky, Cloud transformers: A universal approach to point cloud processing tasks, in: 2021 IEEE/CVF International Conference on Computer Vision (ICCV), 2021, pp. 10695–10704. doi:10.1109/ICCV48922.2021.01054.

[22] H. Zhao, L. Jiang, J. Jia, P. H. Torr, V. Koltun, Point transformer, in: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2021, pp. 16259–16268.

[23] X. Wu, L. Jiang, P.-S. Wang, Z. Liu, X. Liu, Y. Qiao, W. Ouyang, T. He, H. Zhao, Point transformer v3: Simpler faster stronger, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2024, pp. 4840–4851.

[24] Z. Wei, H. Chen, L. Nan, J. Wang, J. Qin, M. Wei, Pathnet: Path-selective point cloud denoising, IEEE Transactions on Pattern Analysis and Machine Intelligence 46 (6) (2024) 4426–4442. doi:10.1109/TPAMI.2024.3355988.

[25] M. Vogel, K. Tateno, M. Pollefeys, F. Tombari, M.-J. Rakotosaona, F. Engelmann, P2p-bridge: Diffusion bridges for 3d point cloud denoising, in: European Conference on Computer Vision (ECCV), 2024.

[26] Z. Liu, Y. Zhao, S. Zhan, Y. Liu, R. Chen, Y. He, Pcdnf: Revisiting learning-based point cloud denoising via joint normal filtering, IEEE Transactions on Visualization and Computer Graphics 30 (8) (2024) 5419–5436. doi:10.1109/TVCG.2023.3292464.

[27] A. Mao, B. Yan, Z. Ma, Y. He, Denoising point clouds in latent space via graph convolution and invertible neural network, in: 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024, pp. 5768–5777. doi:10.1109/CVPR52733.2024.00551.

[28] K. Luan, C. Shi, N. Wang, Y. Cheng, H. Lu, X. Chen, Diffusion-based point cloud super-resolution for mmwave radar data, in: 2024 IEEE International Conference on Robotics and Automation (ICRA), 2024, pp. 11171–11177. doi:10.1109/ICRA57147.2024.10611026.

[29] R. Geng, Y. Li, D. Zhang, J. Wu, Y. Gao, Y. Hu, Y. Chen, Dream-pcd: Deep reconstruction and enhancement of mmwave radar pointcloud, Trans. Img. Proc. 33 (2024) 6774–6789. doi:10.1109/TIP.2024.3512356.
URL https://doi.org/10.1109/TIP.2024.3512356

[30] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, O. Beijbom, Nuscenes: A multimodal dataset for autonomous driving, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 11621–11631.

[31] M. Chamseddine, J. Rambach, D. Stricker, O. Wasenmuller, Ghost target detection in 3d radar data using point cloud based deep neural network, in: Proceedings International Conference on Pattern Recognition, IEEE, 2021, pp. 10398–10403.

[32] T. Griebel, D. Authaler, M. Horn, M. Henning, M. Buchholz, K. Dietmayer, Anomaly Detection in Radar Data Using PointNets, in: Proceedings of the IEEE International Intelligent Transportation Systems Conference (ITSC), 2021, p. 2667–2673. doi:10.1109/ITSC48978.2021.9564730.
URL https://doi.org/10.1109/ITSC48978.2021.9564730

[33] M. Chamseddine, J. Rambach, D. Stricker, O. Wasenmuller, Ghost target detection in 3d radar data using point cloud based deep neural network, in: Proceedings International Conference on Pattern Recognition, IEEE, 2021, pp. 10398–10403. doi:10.1109/ICPR48806.2021.9413247.

[34] Y. Jin, R. Prophet, A. Deligiannis, I. Weber, J.-C. Fuentes-Michel, M. Vossiek, Comparison of different approaches for identification of radar ghost detections in automotive scenarios, in: 2021 IEEE Radar Conference (RadarConf21), 2021, pp. 1–6.

[35] F. Kraus, N. Scheiner, W. Ritter, K. Dietmayer, Using machine learning to detect ghost images in automotive radar, in: 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), IEEE, 2020, pp. 1–7.

[36] W. Wang, R. Yu, Q. Huang, U. Neumann, Sgpn: Similarity group proposal network for 3d point cloud instance segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2569–2578. doi:10.1109/CVPR.2018.00272.

[37] M. Meyer, G. Kuschk, Automotive radar dataset for deep learning based 3d object detection, in: Proceedings of 16th European Radar Conference (EuRAD), 2019, pp. 129–132.

[38] A. Ouaknine, A. Newson, J. Rebut, F. Tupin, P. Pérez, Carrada dataset: Camera and automotive radar with range- angle- doppler annotations, in: Proceedings International Conference on Pattern Recognition, 2021, pp. 5068–5075. doi:10.1109/ICPR48806.2021.9413181.

[39] P. Li, K. Cai, M. R. U. Saputra, Z. Dai, C. X. Lu, OdomBeyondVision: An Indoor Multi-modal Multi-platform Odometry Dataset Beyond the Visible Spectrum, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2022, pp. 3845–3850. doi:10.1109/IROS47612.2022.9981865.

[40] O. Schumann, M. Hahn, N. Scheiner, F. Weishaupt, J. F. Tilly, J. Dickmann, C. Wöhler, Radarscenes: A real-world radar point cloud data set for automotive applications, in: Proceedings of the IEEE 24th International Conference on Information Fusion (FUSION), 2021, pp. 1–8. doi:10.23919/FUSION49465.2021.9627037.

[41] A. Cennamo, F. Kaestner, A. Kummert, Leveraging radar features to improve point clouds segmentation with neural networks, in: Proceedings of the 21st EANN (Engineering Applications of Neural Networks) 2020 Conference: Proceedings of the EANN 2020 21, Springer, 2020, pp. 119–131.

[42] J. Liu, W. Xiong, L. Bai, Y. Xia, T. Huang, W. Ouyang, B. Zhu, Deep instance segmentation with automotive radar detection points, IEEE Transactions on Intelligent Vehicles 8 (1) (2022) 84–94.

[43] J.-T. Huang, C.-L. Lu, P.-K. Chang, C.-I. Huang, C.-C. Hsu, Z. L. Ewe, P.-J. Huang, H.-C. Wang, Cross-modal contrastive learning of representations for navigation using lightweight, low-cost millimeter wave radar for adverse environmental conditions, IEEE Robotics and Automation Letters 6 (2)

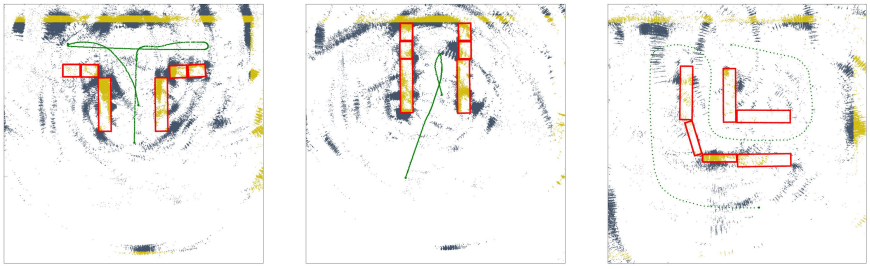
(2021) 3333–3340. doi:10.1109/lra.2021.3062011.
URL http://dx.doi.org/10.1109/LRA.2021.3062011

[44] P. Veličković, Message passing all the way up, arXiv preprint arXiv:2202.11097 (2022).

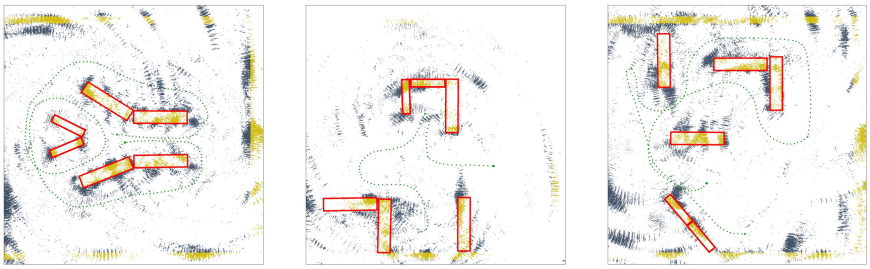
[45] T. Huang, Y. Liu, 3d point cloud geometry compression on deep learning, in: Proceedings of the 27th ACM International Conference on Multimedia, MM '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 890–898. doi:10.1145/3343031.3351061.
URL https://doi.org/10.1145/3343031.3351061

[46] S. Cabello, P. Giannopoulos, C. Knauer, G. Rote, Matching point sets with respect to the earth mover's distance, Computational Geometry 39 (2) (2008) 118–133. doi:https://doi.org/10.1016/j.comgeo.2006.10.001.
URL https://www.sciencedirect.com/science/article/pii/S0925772106001040

Appendix A. MilliNoise Scenarios



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3



(d) Scenario 4 (e) Scenario 5 (f) Scenario 6

Figure A.12: Example runs of each of the six scenes available in the MilliNoise dataset. Obstacles are shown in red. Clean (noise) points are shown in yellow (blue). The trajectory followed by the robot is shown with a dashed line.

To capture several realistic environments – emulating wide and narrow hallways, tight and loose turns, shelves, etc. – we adopted the Texas Instruments' mmWave sensor AWR6843AOP[4], mounted on a differential drive robot (the *Turtlebot3 Waffle Pi*[5]). The collected environments are obtained by arranging diverse boxes (representing obstacles) of different sizes in a 6×6 meters arena. Each arrangement of obstacles in the environment defines a "scenario". For each scenario, the robot follows multiple trajectories (*runs*), capturing the obstacles in the scene from many different points of view.

Together with the mmWave sensor data, the positions of both obstacles and robot have been recorded using the *Vicon Tracker*[6] motion capture system. This system, based on an array of infrared cameras, utilizes passive reflective markers to detect and recognize objects in the scene, providing measurements with an error below a tenth of a millimeter at $\sim 250\,\text{Hz}$.

[4] `https://www.ti.com/product/AWR6843AOP`
[5] `https://www.robotis.us/turtlebot-3-waffle-pi/`
[6] `https://www.vicon.com/software/tracker/`

Data Labelling

MilliNoise labels each point with a sub-millimeter accuracy. In fact, in the post-processing phase, the captured PC frames are used to build a map of the environment. Then, by leveraging the position, provided by the Vicon motion tracking system, and the sizes of obstacles in the map, a label is assigned to each point. In particular, during the labelling process, if a point falls within an obstacle, it is classified as *clean/true*. Conversely, points that do not fall within the spatial region of any obstacle are classified as *noise/false* points.

Appendix B. Computational Complexity

Model	# parameters	training step time (s)	trainins step GPU usage (MiB)	epoch time (s)	epoch GPU usage (MiB)
PointNet	3,552 K	0.05	2050	74	9026
PointNet++	966 K	0.02	4898	99	7060
DGCNN	3,359 K	0.02	2754	113	17,090
Transformer	1,496 K	0.01	2242	85	17,090
GT-*MilliNoise* $\setminus o\text{-}Temp$	1,564 K	0.05	2754	201	10,790
GT-*MilliNoise*	1,632 K	0.08	7362	243	30,420

Table B.10: Computational complexity of the implemented architectures.

In Table B.10, we evaluate the computational complexity of each model. We measure the number of trainable parameters, GPU memory and the computation time for two processing operations: *i)* a training step operation, and *ii)* an epoch step operation. For a training step, the batch size is defined as 1, and the input PC is an accumulation of 12 frames. In the epoch step, the batch size is set to 32 for the same 12 input PC, and the models iterates over the entire training data. As such, given a batch size of 32, an epoch is equivalent to 220 training steps.

The Table shows the computational complexity GT-*MilliNoise* is comparable with state-of-art methods and can be implemented in practical applications. The GT-*MilliNoise* contains $1,632K$ trainable parameters, which in the context of PC processing is considered a moderate-size model. The higher number of trainable parameters of the PointNet and DGCNN models is due to the batch normalization layers. The GT-*MilliNoise* takes 0.08 seconds to perform a training step, only more 0.03 seconds than the commonly used PointNet. It can be seen the GT-*MilliNoise* has significantly higher GPU memory usage when compared with the GT-*MilliNoise* $\setminus o\text{-}Temp$. The reason for this is the implementation of *Temporal* block process each frame in the accumulated input PC *indiviually*. Each interaction performs the closest neighbours search and builds a distance matrix. In the future, this implementation can be greatly optimized to utilize less GPU memory, bringing the computational complexity of the GT-*MilliNoise* closer to the GT-*MilliNoise* $\setminus o\text{-}Temp$ complexity.

Appendix C. GT-*MilliNoise* Architecture Details

Figure C.13 shows the architecture details for the *Geometric* block. Figure C.14 shows the architecture details for the *Temporal* block.

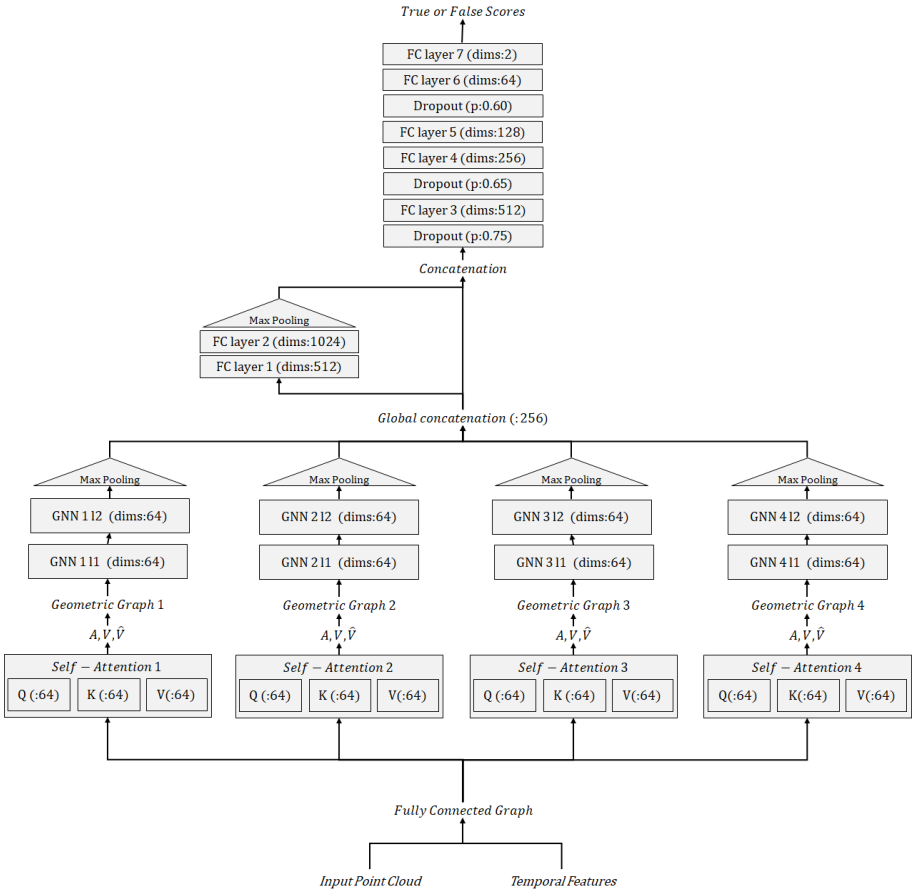


Figure C.13: *Geometric Block - Architecture Details*

Appendix D. GT-*MilliNoise* with *K-hop-GNN* Architecture Details

For the *K-Hop* GT-*MilliNoise* variation, we replace the self-attention operation with *K-Hop Graph* build manually. Figure C.13 shows the architecture details for the *Geometric* block in the *K-Hop* GT-*MilliNoise* variation.

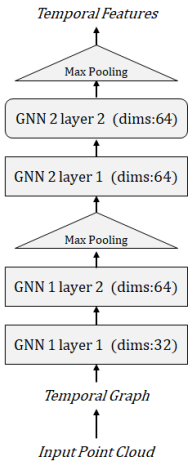


Figure C.14: *Temporal-Block* Architecture Details

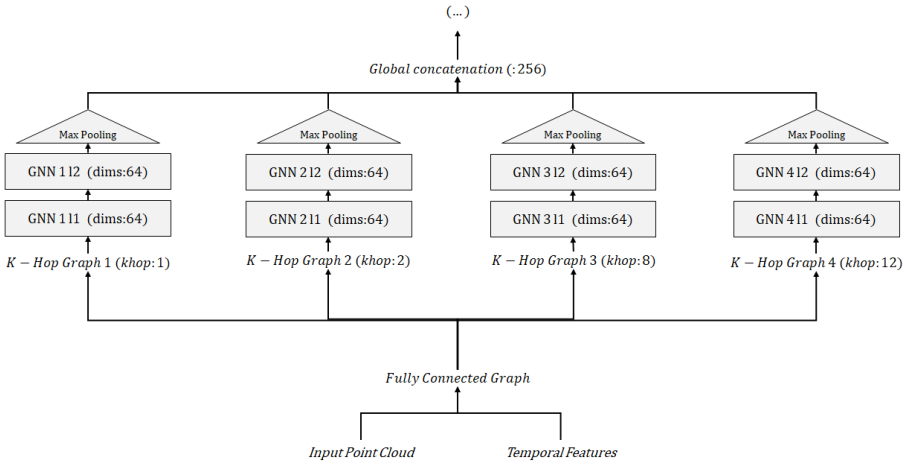


Figure D.15: *Geometric* block Architecture Details - *K-Hop* Variation. The figure is simplified, the final layers can be found in Figure C.13