People Counting from mmWave Radar Point Clouds with Graph Neural Networks

Bernadett Bakos[1]

Supervisors: Marco Zuñiga Zamalloa[1]**, Girish Vaidya**[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfillment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Bernadett Bakos
Final project course: CSE3000 Research Project
Thesis committee: Marco Zuñiga Zamalloa, Girish Vaidya, Michael Weinmann

Abstract

Crowd-control is an emerging problem in urban areas and cameras are commonly seen as the solution, however, there are major concerns regarding privacy. To overcome these issues, while still maintaining the ability to keep track of people, mmWave sensors can be utilized instead, but this introduces new challenges when it comes to people counting. They work by recording the data as point clouds, making it difficult to determine the real number of people when they are occluded. To address this challenge, we combine the spatial and temporal information of the point clouds into graphs, and explore the possibilities of Graph Neural Networks. Our system classifies up to 5 people and bikes with an accuracy of 80.47%. This accuracy is 2.53% worse than the state of the art people counting system for mmWave radar point clouds.

1 Introduction

Our cities are growing at a rapid pace. By 2050, two-thirds of the world's population is expected to live in urban areas [1]. This emphasizes the challenges faced during urban planning and safety measures. To achieve smart planning, people counting, and crowd sensing gives an important metric and are therefore an important assistance. They can be utilized in several cases, such as keeping track of the number of people entering a building or tracking the number of people passing by on the street. This could help prevent overcrowding and gives useful insights about the popularity of certain locations. Further, applications in urban scenarios provide valuable insights into urban planning by showing patterns in traffic and usage trends [2].

There have been several experiments done in the field of people and crowd counting. These can be divided into 2 main categories: vision based and device based. Vision based people counting relies on pictures and videos and therefore uses cameras to collect information. These approaches use deep learning techniques to determine the number of people seen by the camera [3], [4], [5]. Cameras however record identifiable information of the people and are therefore raising privacy concerns. In certain scenarios, such as people counting on the street, respecting people's privacy is of utmost importance. In the European Union, the General Data Protection Regulation (GDPR) specifies in Article 5(1)(c) that the minimum amount of personal data necessary should be collected [6]. For these reasons, visual-based people counting are not always an appropriate option. Device based crowd sensing determines the number of people through their devices. This could include, the Bluetooth signals [7] or the location of these devices. This however requires people to always carry their devices with themselves, and can only give an accurate estimation when everyone does so. To overcome these limitations while preventing privacy issues, mmWave sensors can be utilized for monitoring people.

MmWave radars work by emitting short-wavelength electromagnetic waves and collecting the reflected signals

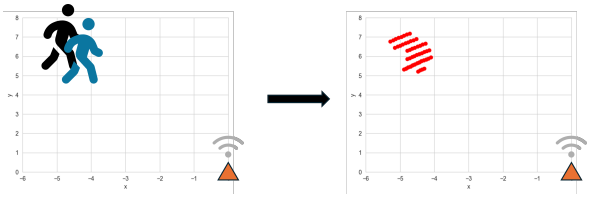


Figure 1: One frame of the mmWave radar recording 2 people as point clouds. The radar is represented with an orange triangle at (0, 0). The people walking in the left image corresponds to the red points in space in the right image.

[8]. The collected signals contain information about the object's range, velocity, and angle. These features can be used to convert the data into point clouds (i.e., a set of data points in 2D space). This representation is visualized in Figure 1, where the people sensed by the radar are represented as points in space and are therefore non-identifiable from the data. As the signals are transmitted with a frequency of 76–81 GHz, the detected movements can be in the range of a fraction of a millimeter. This assures that the sensor has a high accuracy. Moreover, the sensing is independent of weather [9], and time of day, whereas, the performance of cameras can significantly worsen in adverse weather or lighting conditions such as rain or during the night.

As mmWave sensors represent the data as point clouds, it is challenging to determine the number of people detected by the sensor. Figure 1 shows the occlusion caused by 2 people walking together. To address this challenge, there has been previous work done for people counting with the use of deep learning algorithms on mmWave sensor data [10]. This model focuses only on the spatial information of the point clouds, i.e., the frames are independent of each other and their order does not affect the performance of the model. The aim of this paper is to explore the possibility of including the temporal properties, i.e., build a model which takes into account the order of the frames as well as the spatial properties. This approach will be tackled by forming successive frames into graphs and training a Graph Neural Network (GNN) on them.

Our main contributions:

- Temporal Graph Formation algorithm which models points clouds as temporal graphs.
- Performance evaluation of GNNs with temporal graphs for people counting.
- Publicly available code [1]

In this paper, we propose an alternative solution for people counting with GNNs by first explaining the state of the art in Section 2. Then we describe the proposed model in Section 3. Afterward, the evaluation of the proposed architecture is conducted in Section 4. The ethical aspects of the research are discussed in Section 5. A discussion about the results is given in Section 6. Finally, the conclusions are presented in Section 7.

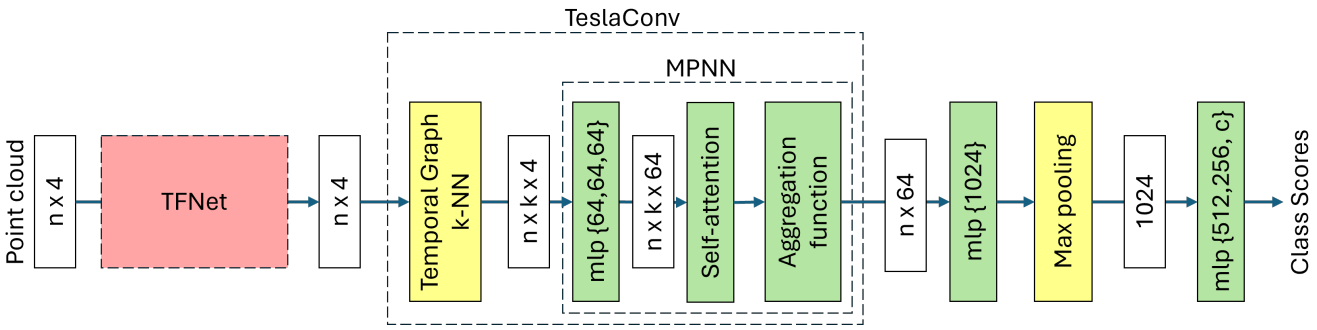


Figure 2: The proposed model, adapted from the Tesla architecture [11]. The elements that are colored with red (TFNet) are excluded, the yellow colored ones (Temporal Graph k-NN, Max pooling) are modified, and the green blocks are kept. Modification for the Temporal Graph K-NN: providing an option between furthest, random, and nearest neighbors. Modification for Max pooling: Average pooling instead of max.

2 Related Work

There are commercially available products for people counting, such as cameras from HIKvision [12] and Canon [13] and sensors from Terabee [14]. These have a high accuracy for people counting, HIKvision and Canon however uses cameras and therefore maintains privacy concerns. The Terabee products uses Time-of-Flight (ToF) sensors, which uses the reflection of light to calculate the distance between points [15], and are therefore more privacy-friendly. Their products are however only for indoors applications currently.

As all current products for people counting come at a trade-off, there is emerging research into alternative solutions, such as LiDAR (utilizes pulsed laser light for distance calculation) and mmWave sensors.

LiDAR provides higher resolution data than ToF, it is however more expensive [16]. Studies have shown that LiDAR sensors provide high accuracy performance for people counting indoors [17]. In comparison to mmWave sensors, they are less resilient to weather conditions and have a shorter range, and therefore are less optimal for outdoor applications. LiDAR sensors are less privacy invasive than cameras, they however still record the human skeletal structure.

MmWave radars provide the least invasive crowd monitoring solution. Recently, this technology has become popular for several human related monitoring tasks, such as people tracking, gesture recognition, human joint estimation, and people counting. These experiments show the potential of mmWave radars. For people tracking, the proposed model [18], using a Deep Recurrent Network, has achieved 89% accuracy for identifying 12 people based on the spatial and temporal properties of the radar point clouds. The Tesla-Rapture model [11] has been proposed for gesture recognition with a 90.53% accuracy, using GNNs on both the spatial and temporal information. MARS [19] was proposed for human joint estimation. It uses a Convolution Neural Network to estimate the location of the joints based on the spatial properties of the point clouds. The estimation performs with an average error of 5.87 cm for all joint positions. With focusing only on the spatial properties of the point clouds, the PointNet architecture [20] achieved 83% accuracy for people counting in outdoor scenarios [10].

As can be seen in the aforementioned experiments, several approaches have been proposed to handle mmWave data. Some of them considering only spatial properties and some both spatial and temporal properties. For people counting, the combination of spatial and temporal properties have not yet been explored. For these reasons, we adapt the proposed Tesla architecture [11] to fit the purposes of people counting. GNNs use graphs as their input, which offers a wide range of possibilities for capturing the temporal information. The relevance of this model is given by having the same type of data with multiclass classification. There are however notable differences between the use cases. We classify into the number of people, while they classify into the different type of gestures. This therefore also means that the datasets are different by nature. Their dataset focuses on the different types of gestures and has mainly a single person standing in front of the radar. Our dataset however represents different number of people walking in front of the radar. A further difference is that they use 3-dimensional point clouds, while our data contains 2-dimensional points.

3 Proposed Model

Inspired by the Tesla architecture [11], the proposed model builds on GNNs, capturing the spatial and temporal properties in the form of graphs. The overview of the model can be seen in Figure 2, where the colors indicate the parts that were kept, modified or excluded in comparison to the Tesla model to adapt for the purposes of people counting.

The Tesla model contains a TFNet (Spatial Transformer Network) component which is responsible for increasing the spatial invariance of the model and therefore making the classification easier. This is a useful component for gesture recognition, since making the same gestures look similar to each other can make the classification more precise. In our case however the same number of people can walk past the sensor in very different shapes, moreover when multiple people pass by as a group, occlusion can occur and the point cloud for different number of people can look very similar. For these reasons, the TFNet would not improve the accuracy in our case and is therefore excluded from the model.

[1]https://github.com/detti456/temporal-graph-neural-network

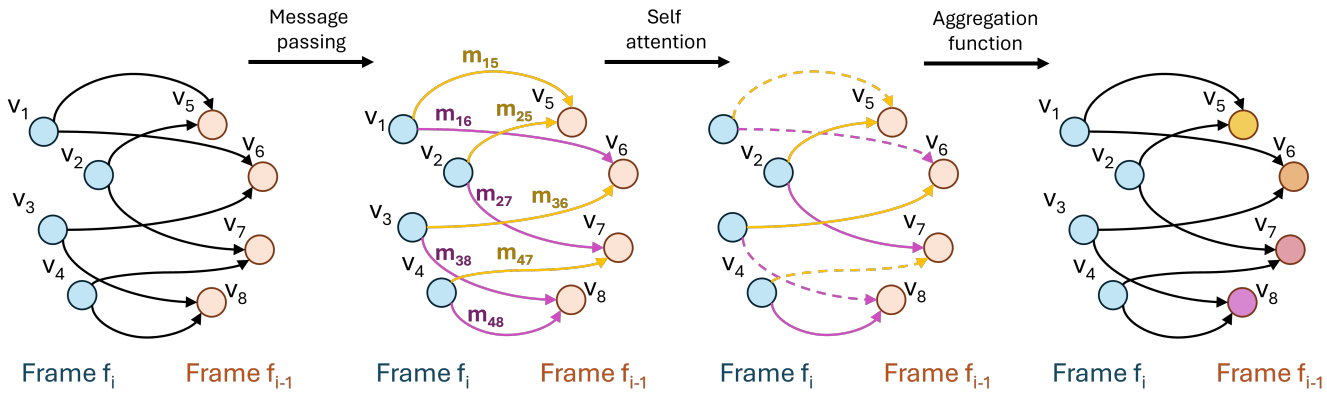


Figure 3: The steps of the graph processing, adapted from the Tesla architecture [11]. First, message passing is utilized where the nodes share information via edges. Second, self attention is applied to determine the importance of the information. Third, the collected information is aggregated at the nodes via an aggregation function. This pipeline shows how the temporal depth of the data is included in the learning process.

The model consists of three main steps. Firstly, data preprocessing is taking place to prepare the data for the GNN. Secondly, the preprocessed data is formed into graphs, to represent the spatial and temporal properties, as the GNN expects graphs as its input. Lastly, the generated graphs get processed through a GNN pipeline.

3.1 Data Preprocessing

The input data consist of 5 features, namely: *Doppler*, *SNR* (signal-to-noise ratio), x, y and *frame number*. The *frame number* works as an identifier for each point to separate the frames from each other. To avoid overlapping frame numbers across different measurement times, the frame numbers are shifted by the maximum frame number from the previous measurement:

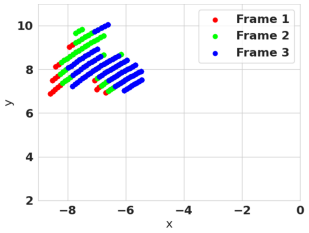
$$f_{di} = \begin{cases} f_{di} & \text{if } d = 0, \\ f_{di} + max(f_{d-1}) + s & \text{otherwise,} \end{cases} \quad (1)$$

where f_{di} is the frame number of the i-th frame on day d, $max(f_{d-1})$ is the largest frame number on the previous day, and s is some constant $s > 3$ to reserve a time gap between two days (as frames from different days should not be successive to each other). This enables to have a unique identifier for all frames, while preserving their relative relationship.

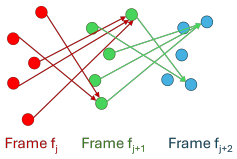
3.2 Graph Generation

The graphs are generated directly from the pre-processed point clouds before passing them into the Neural Network. We opted for a static graph formation mechanism (generating the graphs once, before the learning algorithm) instead of dynamic graph generation (updating the graph structure during training). The reason for this is the fact that we do not use any spatial transformation component (such as TFNet), therefore, the location of the points remain the same in each iteration.

Each graph is composed of f consecutive frames. The order of the frames is determined by the *frame number*, and the difference between the frame numbers represent the time gap between frames. To ensure that all frames in a graph are relevant to each other (i.e., represent the motion of the same object moving), the time gap between any two consecutive frames in one graph should be limited. The frames, that do not have at least $f - 1$ successive frames, satisfying this requirement, are excluded from the data. We set this limit to be at most 8, which is equivalent to skipping 4 frames (two consecutive frames have a time gap of 2). This allows for a larger number of graphs to be created while their relevance is still preserved.



(a) 3 consecutive frames of 5 people



(b) Graph generation from 3 consecutive frames with 1 farthest neighbor

Figure 4: 3 consecutive frames visualized as a point cloud (a) and a simplified example of forming them into a graph (b)

The graphs consist of nodes and edges. The nodes in the graphs represent individual data points from the point clouds, and capture their features (Doppler, SNR, x, y). The frame number is not stored at the nodes, as it is independent of the radar measurements and functions solely as an identifier for the frames.

The edges in the graphs are pointing from each frame to its subsequent frame (leaving the last frame without outgoing edges). Each point p_i in frame f_j are connected to k points from the next frame f_{j+1}. The k connections are determined based on some function F ordering the Euclidean distances between points. The Euclidean distances are calculated for each combination of points (x and y coordinates) in

consecutive frames. These distances are then sorted based on some function F for all points, and the first k distances are formed into an edge. We have used three different functions for F for metrics: sorting in ascending (nearest neighbors) and descending (farthest neighbors) order, and random shuffling. The comparison of these functions can be seen in Section 4.3. An example of graph generation can be seen in Figure 4. Three consecutive frames are plotted as point clouds in Figure 4a and are formed into a simplified graph in Figure 4b. The colored nodes in the graph represent data points from the frame with the matching color. The graph connects each point to its furthest neighbor.

3.3 Graph Processing

The graph processing is composed of three main steps as show in Figure 3: *message passing*, *self-attention* and *aggregation*. These steps are achieved through a Message-passing neural network (MPNN) introduced by Gilmer et al. [21] as shown in Figure 2. This component is responsible for the communication of the nodes through the edges. As the temporal depth of the data is preserved in the edges, it is included in the learning process through MPNN.

The *message passing* and *aggregation* form the basis of MPNN. Therefore, we start by explaining these two concepts. Each node in the graph collects and aggregates the features of the neighboring nodes, as well as the values of the edges, according to:

$$x_i^l = \Gamma_{j \in N(i)}(M_\Phi(x_i^{l-1}, x_j^{l-1}, e_{j,i})), \tag{2}$$

where, x_i^l denotes the features of node i in layer l of the MPNN and $e_{j,i}$ denotes the edge features. Γ represents a differentiable, permutation invariant aggregation function (e.g., sum, mean or max). M_Φ is the message passing function, i.e., a nonlinear function with a set of learnable parameters Φ (here, implemented as Multi Layer Perceptrons (MLP)). This operation is applied to the F-dimensional features of n points (nodes), resulting in the same number of points with dimension F'. Here, F is the number of nodes in a given layer of the neural network, and F' is the number of nodes in the next layer.

The choice of M and Γ makes a significant impact on the performance of the model. For example, setting $M_\Phi(x_i, x_j, e_{j,i}) = \overline{M}_\Phi(x_i)$ captures only the global information of the graph, excluding the local interconnection. This type of message passing function is utilized in the PointNet model. For our proposed model, we opted for an asymmetric function:

$$M_\Phi(x_i, x_j, e_{j,i}) = \overline{M}_\Phi(x_i, x_j - x_i, e_{i,j}). \tag{3}$$

This captures the global information, as well as the local structure of the graph [22]. For a comparison of messaging with edge features ($\overline{M}_\Phi(x_i, x_j - x_i, e_{i,j})$) and without edge features ($\overline{M}_\Phi(x_i, x_j - x_i)$), see Section 4.3. For the aggregation function Γ, we chose *mean*, to preserve information from all nodes.

The message passing is enhanced with a Scaled Dot-Product Multi-Head Self-Attention algorithm [23], which assigns weights to each messages from the neighboring nodes, to determine their importance. First, the messages for each node are concatenated according to:

$$\mathcal{M}_i = \|_{j \in N(i)} \overline{M}_\Phi(x_i, x_j - x_i, e_{i,j}), \tag{4}$$

where $\|$ is the concatenation operator along the first dimension. This results in a matrix representation $\mathcal{M}_i$ of the messages with a shape of $k \times F'$.

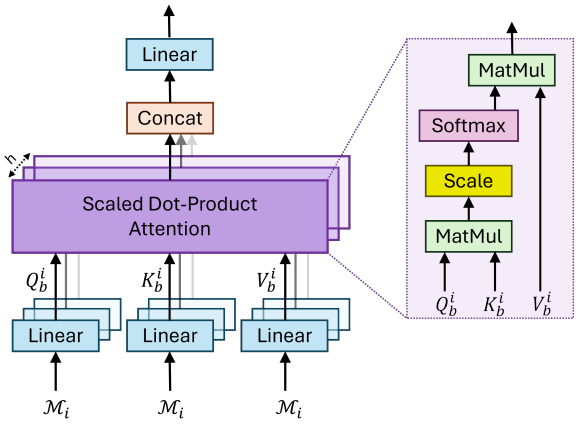


Figure 5: Scaled Dot-Product Multi-Head Self-Attention [23]. Firstly, the *Linear* components project the messages (Equation (5)) in each head. Secondly, the *Scaled Dot-Product Attention* processes the projected messages for each head, as described in Equation (6). Finally, the heads are concatenated and projected to achieve the output (Equation (7)).

The attention mechanism is shown in Figure 5. In case of single-head attention, the algorithm maps queries (Q_b^i), keys (K_b^i), and values (V_b^i) to an output, which all are matrices. In our case, these matrices are formulated by:

$$\begin{aligned} Q_b^i &= \mathcal{M}_i W_b^Q \mid W_b^Q \in \mathbb{R}^{F' \times d_k} \\ K_b^i &= \mathcal{M}_i W_b^K \mid W_b^K \in \mathbb{R}^{F' \times d_k} \\ V_b^i &= \mathcal{M}_i W_b^V \mid W_b^V \in \mathbb{R}^{F' \times d_v}, \end{aligned} \tag{5}$$

where W^Q, W^K, W^V are learned linear projection matrices to d_k, d_k and d_v dimensions, respectively. The projected messages are processed through a Scaled Dot-Product Attention (Figure 5):

$$S_b^i(Q_b^i, K_b^i, V_b^i) = softmax(\frac{Q_b^i (K_b^i)^T}{\sqrt{d_k}}) V_b^i. \tag{6}$$

The query and key matrices are multiplied together, and scaled with $\sqrt{d_k}$. Afterward, softmax is applied to achieve the weights of the values.

This pipeline describes a single-head approach, whereas in our work, we apply a multi-head approach with $h = 8$ heads. This means that the linear projections and Scaled Dot-Product Attention are utilized h times in parallel, with different learned W_b^Q, W_b^K, W_b^V matrices. Multi-head attention allows the model to simultaneously focus on different aspects of the input. Moreover, it allows for a more stable learning

process, since it requires fewer layers than the single-head attention, when applying the same number of projections.

To obtain the output of the attention mechanism the results from the Scaled Dot-Product Attention across all heads are concatenated and projected:

$$A(\mathcal{M}_i) = (\|_{b=1}^{h} S_b^i(Q_b^i, K_b^i, V_b^i))W^O, \quad (7)$$

where $\|$ is the concatenation operator and $W^O \in \mathbb{R}^{hd_v \times F'}$ are learned weights. Therefore, updating the message passing function in Equation (2) to include attention, results in:

$$x_i^l = \Gamma_{j \in N(i)} A(\mathcal{M}_i^{l-1}). \quad (8)$$

4 Experimental Setup and Results

In this section, we evaluate the proposed model. We describe the dataset and its division in Section 4.1. The specifications about the model training are provided in Section 4.2. The impact of the hyperparameters is explored in Section 4.3. Finally, the results are presented in Section 4.4.

4.1 Dataset

The data we evaluate our model on was also used for the classification for people counting with PointNet architecture [10]. It was obtained through a Frequency Modulated Continuous Wave (FMCW) radar, which is a special subclass of mmWave sensors. The data was obtained outdoors and captures people walking in front of the radar. The dataset contains frames of 1, 2, 3, 4 and 5 people walking in front of the sensor as well as bikes passing by. Each frame has been manually classified to one of the aforementioned categories based on a camera recording, to obtain a labelled dataset.

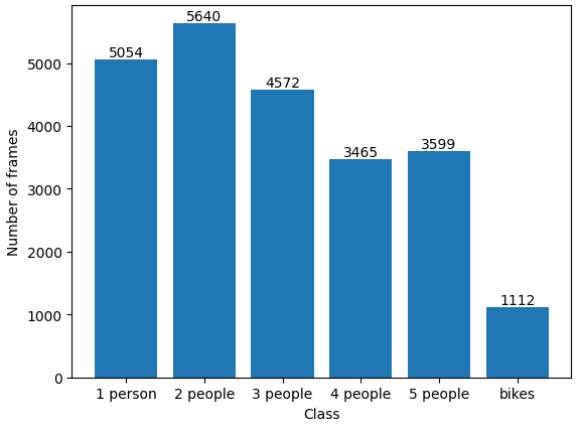


Figure 6: Distribution of the data over the different classes occurring in the dataset.

The direct information collected by the radar includes: range, azimuth (angle), Doppler frequency and signal-to-noise ratio (SNR). From the range and azimuth values, the Cartesian coordinates are calculated with polar to Cartesian conversion:

$$\begin{aligned} x &= r cos(az) \\ y &= r sin(az) \end{aligned} \quad (9)$$

where r is the range and az is the azimuth. This forms the 2D point cloud. Each point in the data is assigned a frame number (increasing over time, but the counter is reset with the device), to preserve the temporal property of the data.

As shown in Figure 6, the data is highly unbalanced. Most frames are captured of 2 people, while 4 and 5 people have more than 2000 frames less than 2 people. Bikes have the least, with only 20% of the maximum number of frames.

The number of frames presented in Figure 6 are however an upper limit on the frames used by the model. In reality, some of these frames get excluded during graph generation, as they contain less than k points (at least k points are required for edge creation). Frames having less than f successive frames are also excluded from the data.

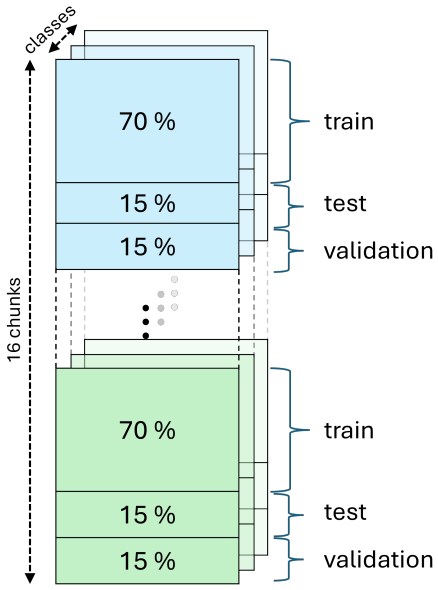


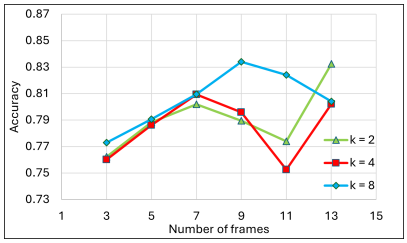
Figure 7: Division of the data into train-test-validation sets, with dividing each classes into 16 chunk and applying the 70:15:15 ratio on each of them.

The data is split into training, testing, and validation sets with a 70:15:15 ratio. As the model requires consecutive frames for graph generation, the data cannot be split randomly. To preserve some variance between the training, testing, and validation sets, the data has been sorted by the frame number. Then, each class is divided into 16 chunks and each chunk, is further divided with a 70:15:15 ratio as illustrated in Figure 7.

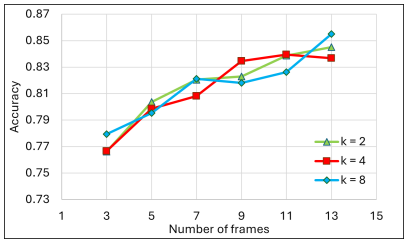
4.2 Model Training

The proposed model has been trained and evaluated on the dataset mentioned in Section 4.1. The training was performed on an NVIDIA GTX 1060 6GB GPU. We implemented the model in Python using PyTorch[2] and PyTorch Geometric[3]. We apply batch-wise training with a batch-size of 32 graphs. The model is trained for at most 50 epochs, with early

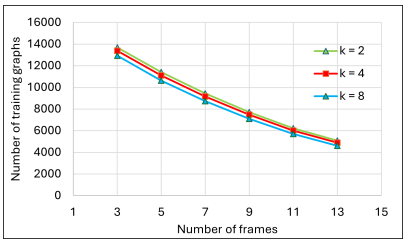
[2]https://pytorch.org/
[3]https://www.pyg.org/



(a) Accuracies with farthest neighbors



(b) Accuracies with random neighbors



(c) Number of training graphs generated

Figure 8: The effect of hyperparameters on farthest neighbors, random neighbors, and number of graphs.

stopping integrated. The learning stops after having no improvement in the performance of the validation set within a patience time of 10 epochs. The model with the best accuracy, on the validation set, is saved. The loss is calculated through a negative log likelihood loss function and the step size is determined by the Adam Optimizer with an initial learning rate of 0.001.

Data Augmentation

To increase the resilience of the model, we use data augmentation. We augment by altering all generated graphs and therefore double the number of graphs used for training. The following augmentations are performed on the generated graphs:

- Random point-wise shifting (jitter) of the points in the graph based on a Gaussian distribution with $\mu = 0$ and $\sigma = 0.1$
- Addition of Gaussian noise to the distance feature of the edges with $\mu = 0$ and $\sigma = 0.1$

4.3 Hyperparameter Tuning

In this section, the performance of the model is evaluated across different selection of hyperparameters. Firstly, we investigate the impact of the distance feature in the message passing step, together with the different edge creation functions. Secondly, we explore the impact of the number of frames and edges in the graphs.

	With distance	Without distance
Nearest neighbors	67.72%	69.61%
Farthest neighbors	79.11%	77.73%
Random neighbors	81.41%	80.75%

Table 1: Accuracy achieved for different edge creation method and usage combinations. The rows represent the different edge generation functions, and the columns provide the edge feature inclusion and exclusion from message passing.

We experiment with different combinations of edge generation functions, and distance inclusion to determine how the edge creation and usage in the graphs influences the model. The included edge generation functions are *nearest*, *farthest*, and *random* neighbors. Each of these functions are combined with distance inclusion or exclusion from the message passing, resulting in six different model settings. The model has been trained for each setting, with $f = 6$ frames and $k = 4$ neighbors in each graph.

The trained models are evaluated on the validation set. The resulting accuracies are presented in Table 1. Here, we can see that the graphs, created by the nearest neighbors policy, perform 10% worse in accuracy, both in case of distance inclusion and exclusion. Random neighbors outperform both nearest neighbors (by around 14%) and farthest neighbors (by around 2%). When looking at the usage of the edges, we can see that including the edge feature (distance) in the training, improves the classification in case of farthest and random neighbors by around 1% and decreases in case of nearest neighbors by around 2%.

In the aforementioned results, we can observe that the performance of nearest neighbors are significantly worse than the others. The reason for this is that most of the frames have overlapping points. This means that some points in a frame are still present in the next frame in the exact same location. With the nearest neighbors connection, these points get connected to each other with a distance of 0 and therefore provide less useful information in the classification process. Connecting frames to farthest neighbors create graphs that are prone to emphasize noise. Moreover, this approach will create edges that are connecting to the outside of the shape, leaving the inside without edges. Whereas random neighbors by definition connect random points and, therefore, ensure a more diverse inclusion into the edge system. We can see that the inclusion of the distance between points gives a slight increase in the accuracy, except for nearest neighbors, where a significant amount of edges have a distance of 0.

Since farthest and random neighbors have a similar performance in case of graphs with $f = 6$ frames and $k = 4$ neighbors, these two edge formation mechanisms are further explored with different number of frames and edges. The distance is included in the training process for all settings.

Figure 8a and 8b shows the effect of frames and edges in terms of accuracy. We can see that the performance of the model is increasing with the number of frames in case of random neighbors (Figure 8b). The number of neighbors and transitively the number of edges however do not have a significant impact on the performance. On the contrary, for farthest neighbors (Figure 8a), the performance is increasing until 7 frames and turns inconsistent afterward. No trend can be observed in terms of number of neighbors. These observations prove that a more diverse edge inclusion leads to a more accurate classification when applied to more than 7

consecutive frames.

The largest amount of frames, considered, was 13. The reason for this is the fact that the more frames are included in a graph, the less graphs can be generated from the data (due to graph formation requirements). This relation is depicted in Figure 8c. Here we can see a 66% drop in the number of graphs between 3 and 13 frames. It is important to note however that the number of graphs doesn't directly translate to the number of frames (as each graph is composed of multiple frames). This is especially relevant for bikes, as they are largely underrepresented in the data (Figure 6). Including more than 13 frames in the graphs would lead to having little to no data representing the bikes.

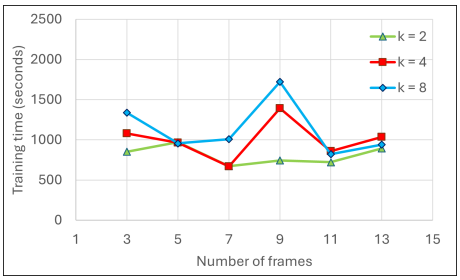


Figure 9: Amount of time taken for training the model, with random neighbors

Taking the aforementioned reason into account, we chose $f = 11$ as the optimal number of frames in combination with random neighbors. The model performs well under this condition while still maintaining enough information about all classes. Since the number of edges does not impact the accuracy of the model, we chose the number of neighbors based on the running time of the model. In Figure 9, we can see that the time taken to train the model decreases with the number of edges. Therefore, we chose $k = 2$ neighbors for each point.

4.4 Results

For the final evaluation we chose the model to have $f = 11$ number of frames, $k = 2$ number of random neighbors and inclusion of the distance feature, based on the observations made in Section 4.3. We trained the model 10 times and evaluated it on the testing set, achieving an average accuracy of **80.47%** with a standard deviation of 1.93%. Comparing these results to the 83% accuracy of the PointNet architecture [10] we can observe a 2.53% decrease in accuracy.

The confusion matrix can be seen in Figure 10. Here, we can observe that the model performs the best in case of 5 people and bikes. The bikes tend to have a different shape and speed in comparison to the walking people, which could be the reason for the better classification. We can also see that the model is more prone to overcounting than undercounting. Almost 66% of the mistakes were made by overcounting. This also explains the high accuracy for 5 people, as there is no other class with a larger amount of people. The most misclassification happened for 4 people, where 23% got misclassified as 5 people and 14% as 3 people. Furthermore, we can observe that most mistakes were made

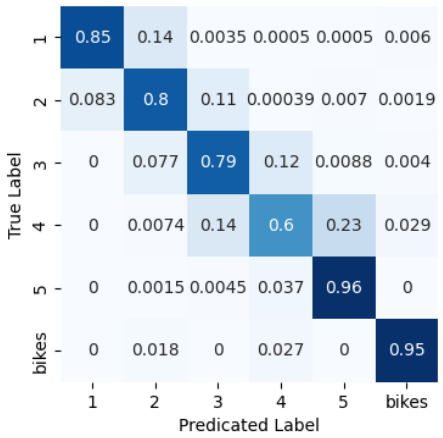


Figure 10: Confusion matrix of the evaluated model for $k = 2$ and $f = 11$

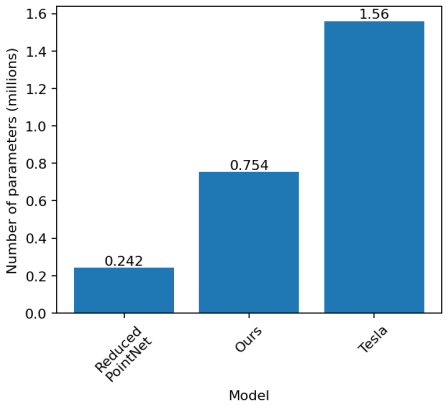


Figure 11: Comparison of our model size with the reduced PointNet and the original Tesla model

with neighboring classes, meaning that the model is confused with $\pm$ 1 person.

The proposed model has a size of 2.890MB. The number of parameters is compared with the reduced PointNet [10] and the original Tesla model in Figure 11. Our model is reduced in size in comparison to the original Tesla model, due to the exclusion of some components, mentioned in Section 3. It is however still larger than the reduced PointNet.

5 Responsible Research

In Section 5.1, we discuss the collection and diversity of the data. The reproducibility of the results are presented in Section 5.2. Finally, Section 5.3 explains the ethical aspects of the mmWave radars.

5.1 Dataset

The data was collected for people counting with the PointNet architecture. A group of five volunteers walked in front of the radar in different combinations. The collected data has been manually classified by volunteers, based on a camera recording, to obtain a labelled dataset. After the labeling, the

data does not contain any identifiable information and is used in this form by our model. The people count is stored together with a rough position of the people without any recognizable human skeletal structure (only sparse points in space). This dataset is not publicly available. As the dataset is composed of only five different people, the capability of the model to generalize for new people is unknown.

As the proposed model requires consecutive frames, to avoid reusage of frames from the training, we take chunks of successive frames for testing and validation. These frames are not involved in the training anyhow.

5.2 Reproducibility

The graph generation component contains steps which use random numbers. To achieve reproducibility for these steps, we have set a seed each time a random number is generated. By using the same seed, the random number generator will always return the same number. For further reproducibility, we have made the code public.

On the contrary, due to the nature of neural networks, two different models cannot be trained the same. To reduce the impact of this, we have trained the model 10 times and reported the average of the results as a final result.

5.3 Application

The aim of the paper is to propose a solution for people counting, that performs well in outdoors scenarios. MmWave radars do not collect any identifiable data of the people when utilized individually. However, when it comes to constant monitoring, by placing radars after each other, a walking pattern of individuals can be tracked. To this end, it is important to avoid aggregation of the data.

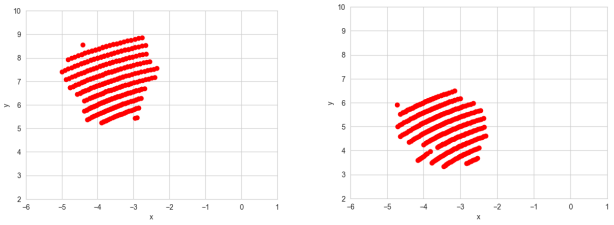
6 Discussion

The proposed model performs worse than the PointNet architecture [10] by 2.53%, showing that the inclusion of the temporal information through the proposed methodology, does not help the classification process.

There are several differences between the two models in the data usage aspect. Firstly, the PointNet architecture applies a random splitting into training, testing and validation sets, while we divide the data based on sorted chunks as explained in Section 4.1. This leads to potentially having more unseen data in the testing and validation sets, as these contain frames that are not related to the training frames anyhow. Whereas, when utilizing random splits, a test frame might be consecutive to a training frame (and therefore have a more similar structure).

Secondly, The PointNet architecture, uses all the available data in the training and evaluation process. We however had to exclude a significant part of the data due to the graph generation requirements (frames not having enough consecutive frames or frames having less than k points). Our used data is therefore significantly less than the original amount of data. For graphs generated with $f = 11$ number of frames and $k = 2$ number of neighbors, around 1500 frames are excluded from the classification.

As discussed in Section 4.4, misclassification is common between consecutive number of people. This could be due to the similar representation of different number of people walking together. This similarity can also be observed in Figure 12, where the point cloud of 3 and 4 people in 5 consecutive frames are plotted. Due to occlusion, the different number of people are still represented as one cluster.



(a) 5 consecutive frames of 3 people

(b) 5 consecutive frames of 4 people

Figure 12: Point clouds of different number of people in front of the radar

7 Conclusions and Future Work

In this work, we proposed a model for people counting from mmWave radar point clouds. The proposed model includes both spatial and temporal information of the data. The point clouds are formed into graphs and processed through a MPNN. We have achieved an accuracy of 80.47% which performs 2.53% worse than the state of the art PointNet architecture. Further enhancements can be made by training the model on a larger and more diverse dataset, which has longer sequences of consecutive frames. Moreover, further edge formation mechanisms, such as deep learning algorithms, can be explored to achieve a dynamic graph generation.

References

[1] United Nations Human Settlements Programme, "Urban Planning | UN-Habitat." [Online]. Available: https://unhabitat.org/topic/urban-planning

[2] Fabrique, "Privacy preserving crowd sensing systems [longread]." [Online]. Available: https://www.ams-institute.org/news/privacy-preserving-crowd-sensing-systems-longread/

[3] X. Guo, M. Gao, G. Zou, A. Bruno, A. Chehri, and G. Jeon, "Object Counting via Group and Graph Attention Network," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–12, 2023, conference Name: IEEE Transactions on Neural Networks and Learning Systems. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10342885

[4] C. Zhang, Y. Zhang, B. Li, X. Piao, and B. Yin, "CrowdGraph: Weakly supervised Crowd Counting via Pure Graph Neural Network," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 20, no. 5, pp. 135:1–135:23, Jan. 2024. [Online]. Available: https://dl.acm.org/doi/10.1145/3638774

[5] U. Bhangale, S. Patil, V. Vishwanath, P. Thakker, A. Bansode, and D. Navandhar, "Near Real-time Crowd Counting using Deep Learning Approach," *Procedia Computer Science*, vol. 171, pp. 770–779, Jan. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S187705092031053X

[6] "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance)," Apr. 2016, legislative Body: EP, CONSIL. [Online]. Available: http://data.europa.eu/eli/reg/2016/679/oj/eng

[7] J. Weppner and P. Lukowicz, "Bluetooth based collaborative crowd density estimation with mobile phones," in *2013 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, Mar. 2013, pp. 193–200. [Online]. Available: https://ieeexplore.ieee.org/document/6526732

[8] C. Iovescu and S. Rao, "The fundamentals of millimeter wave radar sensors," Tech. Rep., 2020.

[9] Texas Instruments, "Can mmWave see in the rain?" Oct. 2017. [Online]. Available: https://www.youtube.com/watch?v=qylLiJaOVqI

[10] G. Vaidya and M. Zuniga, "Exploiting mmWave and Deep-Learning Models to Estimate People Count in Urban Scenarios."

[11] D. Salami, R. Hasibi, S. Palipana, P. Popovski, T. Michoel, and S. Sigg, "Tesla-Rapture: A Lightweight Gesture Recognition System From mmWave Radar Sparse Point Clouds," *IEEE Transactions on Mobile Computing*, vol. 22, no. 8, pp. 4946–4960, Aug. 2023, conference Name: IEEE Transactions on Mobile Computing. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9720163

[12] "People Counting." [Online]. Available: http://www.hikvision.com/europe/solutions/solutions-by-function/people-counting/

[13] C. Europe, "Crowd People Counter - Business Software." [Online]. Available: https://www.canon-europe.com/business/products/software/crowd-people-counter/

[14] "GDPR People Counting | People traffic counters (98% Accuracy)." [Online]. Available: https://www.terabee.com/products/people-counting/

[15] Terabee, "Time-of-Flight principle (ToF): Brief overview, Technologies and Advantages," Feb. 2022. [Online]. Available: https://www.terabee.com/time-of-flight-principle/

[16] Kate, "Time of Flight Sensor vs. LiDAR: What Are the Differences?" Mar. 2024. [Online]. Available: https://pmt-fl.com/time-of-flight-sensor-vs-lidar-what-are-the-differences/

[17] A. Günter, S. Böker, M. König, and M. Hoffmann, "Privacy-preserving People Detection Enabled by Solid State LiDAR," in *2020 16th International Conference on Intelligent Environments (IE)*, Jul. 2020, pp. 1–4, iSSN: 2472-7571. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9154970

[18] P. Zhao, C. X. Lu, J. Wang, C. Chen, W. Wang, N. Trigoni, and A. Markham, "mID: Tracking and Identifying People with Millimeter Wave Radar," in *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, May 2019, pp. 33–40, iSSN: 2325-2944. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8804831

[19] S. An and U. Y. Ogras, "MARS: mmWave-based Assistive Rehabilitation System for Smart Healthcare," *ACM Transactions on Embedded Computing Systems*, vol. 20, no. 5s, pp. 1–22, Oct. 2021. [Online]. Available: https://dl.acm.org/doi/10.1145/3477003

[20] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI: IEEE, Jul. 2017, pp. 77–85. [Online]. Available: http://ieeexplore.ieee.org/document/8099499/

[21] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural Message Passing for Quantum Chemistry," Jun. 2017, arXiv:1704.01212 [cs]. [Online]. Available: http://arxiv.org/abs/1704.01212

[22] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic Graph CNN for Learning on Point Clouds," Jun. 2019, arXiv:1801.07829 [cs]. [Online]. Available: http://arxiv.org/abs/1801.07829

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," Aug. 2023, arXiv:1706.03762 [cs]. [Online]. Available: http://arxiv.org/abs/1706.03762