Salami, Dariush; Hasibi, Ramin; Palipana, Sameera; Popovski, Petar; Michoel, Tom; Sigg, Stephan

Tesla-Rapture

Published in:
IEEE Transactions on Mobile Computing

DOI:
10.1109/TMC.2022.3153717

Published: 01/08/2023

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Please cite the original version:
Salami, D., Hasibi, R., Palipana, S., Popovski, P., Michoel, T., & Sigg, S. (2023). Tesla-Rapture: A Lightweight Gesture Recognition System from mmWave Radar Sparse Point Clouds. *IEEE Transactions on Mobile Computing*, *22*(8), 4946-4960. https://doi.org/10.1109/TMC.2022.3153717

Tesla-Rapture: A Lightweight Gesture Recognition System from mmWave Radar Sparse Point Clouds

Dariush Salami*, Ramin Hasibi*, Sameera Palipana, Petar Popovski, Tom Michoel, and Stephan Sigg

Abstract—We present Tesla-Rapture, a gesture recognition system for sparse point clouds generated by mmWave Radars. State of the art gesture recognition models are either too resource consuming or not sufficiently accurate for the integration into real-life scenarios using wearable or constrained equipment such as IoT devices (e.g. Raspberry PI), XR hardware (e.g. HoloLens), or smart-phones. To tackle this issue, we have developed Tesla, a Message Passing Neural Network (MPNN) graph convolution approach for mmWave radar point clouds. The model outperforms the state of the art on three datasets in terms of accuracy while reducing the computational complexity and, hence, the execution time. In particular, the approach, is able to predict a gesture almost 8 times faster than the most accurate competitor. Our performance evaluation in different scenarios (environments, angles, distances) shows that Tesla generalizes well and improves the accuracy up to 20% in challenging scenarios, such as a through-wall setting and sensing at extreme angles. Utilizing Tesla, we develop Tesla-Rapture, a real-time implementation using a mmWave Radar on a Raspberry PI 4 and evaluate its accuracy and time-complexity. We also publish the source code, the trained models, and the implementation of the model for embedded devices.

Index Terms—Gesture-recognition, Machine-learning, Sensing, Graph-convolution, mmwave radar

1 Introduction

Human computer interaction systems, such as smart home, vehicular, or human-robot interaction, prominently utilize gesture recognition [1], [2], [3]. While classical systems rely on ultra-sound [4], [5], IMU [6], [7], or camera sensors [8], [9], recent approaches have exploited electromagnetic radiation, such as Channel State Information (CSI) or radar, to address the limited sensing range (e.g. untrasound), discomfort of wearing (IMU), occlusion (lidar) or risk of privacy leakage (RGB-depth). In particular, mmWave radar sensing is capable of fine-grained movement recognition, robust to environmental lighting or weather conditions, and can penetrate thin, non-metallic surfaces. Additionally, in monostatic operation, it is capable of providing 3D spatial information through multi-antenna systems. The high operating frequencies allow for small form factors so that the sensor can be mounted on miniature devices. Millimeter waves are non-ionizing and thus not dangerous to the human body.

The input representation plays an important role in both accuracy and time-complexity of deep learning based systems (RGB images [10], [11], [12], depth images [10], [11], [12], spectrograms of Doppler signals [13], and point clouds [14], [15], [16], [17]).

In particular, converting the raw Analog to Digital Conversion (ADC) data from the antenna arrays to point clouds (i.e., unordered sets of points in space), massively reduces the data size by several magnitudes (e.g. GBytes to MBytes), which results in faster data transfer, pre-processing, and inference time. Unlike spectrograms of Doppler signals, point clouds are easily interpretable since the motions occur in a 3D space. Furthermore, strong point-cloud processing models exist, since this format is the standard output of a wide range of sensors [14].

We distinguish these point cloud processing models into multi-view (projection into 2D-planes for 2D processing) [18], [19], volumetric (features generated from voxels in 3D space) [20], [21], and direct, permutation invariant point cloud interpretation (without intermediate representation) [14], [22]. In contrast to direct processing of point clouds, multi-view and volumetric techniques are lossy (reduced accuracy) and computationally intensive (time consuming).

To achieve a low complexity model, we therefore propose a direct point cloud processing method, Tesla (**TE**mporal graph **S**e**L**f **A**ttention convolution), a Message Passing Neural Network (MPNN) graph convolution based architecture tailored to sparse point clouds generated by mmWave radars. Utilizing the unique properties of mmWave radar point clouds, we introduce a novel Temporal Graph K-Nearest Neighbor (K-NN) algorithm to dynamically model the temporal evolution of the point cloud over successive frames as a graph structure, and a novel self-attention MPNN based graph convolution layer called TeslaConv to process the generated graph and infer the gestures. Unlike Recurrent Neural Network (RNN)

- *D. Salami, S. Palipana, and S. Sigg are with the Department of Communications and Networking, Aalto University, Espoo, Finland. E-mails: {dariush.salami, sameera.palipana, stephan.sigg}@aalto.fi*
- *R. Hasibi and T. Michoel are with the Department of Informatics, University of Bergen, Bergen, Norway. E-mails: {ramin.hasibi, tom.michoel}@uib.no*
- *P. Popovski is with the Department of Electronic Systems, Aalborg University, Aalborg, Denmark. E-mail: petarp@es.aau.dk*

* *Both authors contributed equally to this research.*

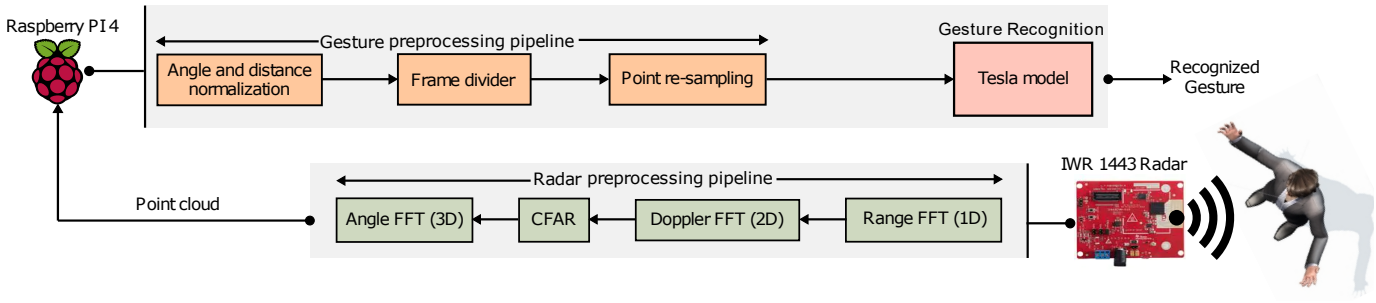


Fig. 1. Overview structure of Tesla-Rapture. The radar transforms the IQ samples into a point cloud through the radar processing pipeline and this is fed to a Raspberry Pi 4 for further processing and infer the gestures.

based models, which iteratively fuse spatial features of each time frame, our method takes advantage of a novel graph convolution with a single forward pass to capture the temporal evolution by reflecting the temporal evolution of the gesture in the intermediate graph structure. As a result, this approach outperforms the state of the art in terms of accuracy and computational complexity, which positions it for embedded devices and real-time settings. In particular, Tesla is ahead of the state of the art by a margin of up to 4.2%, 2.9%, and 9.09% on primary settings as well as 21% in challenging scenarios of three different datasets. It is important to emphasize that the model is 8 times faster and has almost 40 times less computational complexity than the most accurate competitor when it comes to inference time and Giga Floating Point OPerations (GFLOPs), respectively.

Given the widespread usage of Raspberry PI in the Internet of Things (IoT) from human-robot interaction [23], [24] to smart-home applications [25], [26], we integrate Tesla in a system called Tesla-Rapture (**Tesla** for **RA**dar generated **P**oint cloud ges**TURE**) which can be executed on a Raspberry PI 4. The architecture is depicted in Fig. 1.

Our main contributions are:

- Temporal Graph K-NN, a novel Graph K-NN algorithm to model the time dimension of point clouds as a temporal graph.
- This is the first work that processes motion point clouds using a graph convolution approach and develops a self-attention MPNN to process the temporal graph built through the Temporal Graph K-NN.
- A thorough performance evaluation on three datasets with different settings including diverse environments, distances, angles and speeds.
- An implementation on a Raspberry PI 4 in a real-time setting.
- Publicly available code [1], trained models, and Raspberry PI implementation for verification and follow-up research purposes.

2 Related Work

2.1 Gesture Recognition

RGB cameras, RGB depth sensors, Leap Motion, mmWave radars, and WiFi are prominently utilized in the literature for mid-air gesture recognition. Extensive surveys on vision-based gesture sensing were published by Wachs et al. [27] and Rautaray et al. [28]. These systems (e.g. MS Kinect) employ an RGB camera and an infrared depth sensor providing either 2D color frames, full-body 3D skeleton, or 3D point clouds [29]. However, they are limited in darkness and occlusion, and the camera raises privacy concerns [30].

Radio Frequency (RF) gesture recognition can be distinguished into sub-6 GHz and millimeter waves. The former leverages received signal strength [31] from commodity narrow-band devices, CSI from WiFi [32], [33], [34], [35], Doppler [36], or radar [37], [38]. However, the gesture recognition accuracy below 6 GHz is limited by its small bandwidth and a wavelength above 5 cm, so that antenna array apertures become too large. In contrast, mmWave sensing features high bandwidth (4-7 GHz) and smaller antenna apertures. For mmWave radars, gesture recognition is either model [39], [40] or data-driven [41], [42], [43], [44], [45], [46]. Most data-driven approaches combine Convolutional Neural Network (CNN) and RNN modules to process Doppler, range-Doppler, and/or angle-Doppler features [41], [42], [44]. Since these features are dependent on relative direction of movement and angle granularity, complex tasks, such as distinguishing simultaneous movement of different body parts becomes challenging.

Furthermore, although there exist systems that directly process I/Q data [47] or Doppler spectrograms [42], the amount of data generated by the radar is in the order of GBs per second for both I/Q and Doppler spectrograms, which makes it a challenge to transfer and process in real-time on embedded devices, such as Raspberry PI.

2.2 Static Point Clouds

Point clouds from RGB-depth images, LiDAR or mmWave radars differ in their granularity. While point clouds extracted from RGB-depth images and LiDAR are dense, mmWave radars produce sparser point clouds [48] that do not highlight the human skeletal structure [16]. Recent years have witnessed the emergence of mmWave radar point cloud human sensing due to the availability of commercial hardware that is miniaturized and low cost (e.g. hand tracking [49], gesture recognition [16], [50] activity recognition [43], gait recognition [46], or positioning [45]).

PointNet [14], the pioneering model for direct processing of 3D point clouds extracts features on a point-by-point basis and aggregate them using permutation-invariant pooling.

1. https://github.com/dariush-salami/attention-based-edge-convolution

In *PointNet++* [51], set abstraction modules to sample and group neighbouring points in each processing layer have been added to for improved representation of spatial features.

At the same time, by applying CNNs on graphs, graph convolution approaches have emerged [52]. By modeling point clouds as graphs, in which nodes correspond to points, connected to their nearest neighbours through edges in an Euclidean space, graph convolution principles can be applied. In particular, a message passing algorithm (MPNN), is utilized to gradually propagate each point's features as a message to its neighbours and to aggregate the incoming messages with the features of the point itself [53]. Based on MPNN, *Dynamic Edge Convolution (DEC)* [54] redefines the graph through nearest neighbour operations at each convolution layer and messages as Euclidean distance between neighbouring points. Although DEC performs well on shape classification, it fails to capture temporal dependency in mid-air gesture recognition. This shortcoming specifically affects gestures that differ mainly in their temporal, but not in their spatial distribution (i.e. similarity in time-aggregated point clouds, e.g. opposing gestures such as swipe-left and swipe-right). To address this issue, we reflect the temporal evolution of gestures in the graph structure. In particular, each point can only connect to points from previous frames.

2.3 Dynamic Point Clouds

Previous attempts to capture spatio-temporal features of dynamic point clouds include using a combination of RNN with either 3DCNN or PointNet layers [16], [17], [55], as well as using a modified RNN layer to propagate information temporally while preserving the spatial structure in each frame [15]. In real-world applications, these models are constrained by their high computational complexity and restricted generalizability on point clouds generated in different settings. However, given the sparsity of the mmWave radar point clouds in each frame (in average 5-10 points per frame), extraction of frame-wise spatial features does not contribute to the latent representation of gestures. Moreover, the recurrent pipeline of RNN-based model increases the computational complexity. To tackle this problem, we capture the temporal dependency reflected in the graph structure using a single pass of the proposed MPNN model. To further increase the performance of the model, we integrate the self-attention mechanism [56] to increase the impact of important parts of the input data while fading out the rest.

3 Point Clouds from mmWave Radars

We use point cloud datasets from the Texas Instruments IWR1443[2] sensor, a Frequency-Modulated Continuous Wave (FMCW)–Multiple-Input and Multiple-Output (MIMO) radar sensor that operates in the 77 GHz RF band. A radar transmitter antenna (Tx) emits an electromagnetic signal, which is reflected and scattered by objects in the environment, before it is captured again by a receiving antenna (Rx). An FMCW signal is used for range estimation of the reflecting objects and a MIMO configuration is utilized to compute both elevation and azimuth angles [16]. A coordinate transformation of the range, azimuth and elevation angles of the detected objects yields the point cloud in a x-y-z coordinate system. The following signal processing pipeline achieves the point cloud representation from ADC data.

2. https://www.ti.com/product/IWR1443

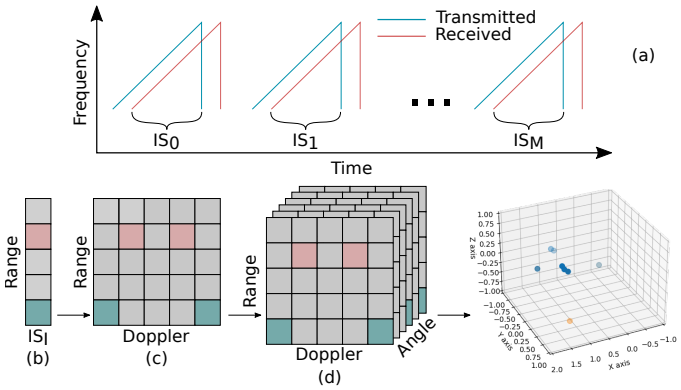


Fig. 2. (a) The transmitted and the reflected chirps are shown in the frequency domain. (b) The range of the detected objects after applying 1D-FFT on the intermediate signal. (c) The velocity of the detected objects after 2D-FFT. (d) The angle of the detected objects after applying 3D-FFT on the data from multiple antennas

3.1 Point Cloud Generation

The processing unit on the evaluation kit of the radar applies a four step preprocessing pipeline to obtain point clouds.

Range-FFT (1D): The radar sends a chirp signal (Fig. 2.a), i.e, a signal with linearly increasing carrier frequency, and produces an intermediate frequency signal by mixing the transmitted and received chirps and low pass filtering. The distance to the reflecting object is proportional to the intermediate frequency, which is computed using the FFT operation on the mixed signal (Fig. 2.b).

Doppler-FFT (2D): Two or more time-separated chirps are required to estimate the radial velocity of an object. The phase difference between two chirps at the range-FFT peak is proportional to the radial velocity of the detected object (2D-FFT or Doppler-FFT) which is shown in Fig. 2.c.

Constant False Alarm Rate (CFAR): The CFAR detection algorithm [57] is used to separate reflecting objects from noise. The summation of the Doppler-FFT matrices creates a pre-detection matrix. The CFAR algorithm identifies peaks in the pre-detection matrix that correspond to the detected objects. The elements with gray color in Fig. 2 show the noisy points that are filtered by CFAR algorithm.

Angle-FFT (3D): For each object in the CFAR algorithm, an FFT of the angle is performed on the corresponding CFAR peaks across multiple Doppler-FFTs (Fig. 2.d). Velocity-induced phase changes are Doppler-corrected before computing the angle-FFT.

The aforementioned process results in points that have 3D coordinates with respect to the position of the radar, time sequence, and intensity of the reflection. The point cloud generated by the radar is sparse since the CFAR algorithm detects peaks in the pre-detection matrix as discussed before. If we reduce the threshold in CFAR to detect more

peaks, the false alarm rate will increase resulting in noisy point clouds.

3.2 Point Cloud Properties

The point cloud generated from the above process has unique spatial and temporal properties.

3.2.1 Spatial properties

The point cloud is sparse and the skeleton structure of the human is not apparent in individual frames. The radar captures more points during motion than during stationary phases of an object or subject. This is attributed to the signal processing tool chain used for the radar. First, the point cloud is extracted through range-FFT, Doppler-FFT, CFAR, and angle-FFT operations as described above. The CFAR algorithm relies on range and Doppler dimensions to detect an object, so that the detected cloud points are triggered due to the motion and intensity of the reflection. This property is used to filter stationary reflections in the environment.

The gestures in the horizontal plane have a higher granularity than in the vertical plane, since the radar has more antenna elements in the azimuth direction. Eight virtual elements can resolve an angle of $14.3°$, in contrast to only $57°$ via two virtual antennas in the elevation direction. Another reason is the sensitivity of the CFAR algorithm in the Doppler direction.

For reflecting objects or subjects close to the sensor, the larger radar cross-section results in denser point clouds. For instance, representations of arms or hands become less sparse in short distance case. Additionally, for reflections off objects in a distance D, the spacing between points captured at a resolution of θ is proportional to $D \cdot \theta$. This causes the point cloud to have a distance-dependent density and causes the trained model accuracy to deteriorate with increasing distance.

3.2.2 Temporal properties

The CFAR algorithm collapses points that are detected over a specific fixed temporal duration t_Δ into frames. The number of cloud points is variable across frames. Even though the skeletal structure of the body is not apparent in individual frames, an arm's motion constructs a spatio-temporal structure in the direction of motion over successive frames. These unique spatio-temporal structures in the point cloud for different gestures can be exploited for motion gesture recognition.

3.3 Comparison with RGB-D Point Clouds

Compared to RGB-D point clouds, mmWave point clouds are sparse. We illustrate this in Fig. 3 using similar gestures from two datasets. In particular, we utilize the Upper Body Point Cloud Gestures (UBPG) RGB-D gesture point cloud dataset [55], and the Pantomime dataset [16] (point clouds of gestures captured by a mmWave radar). Indeed, mmWave point clouds hold little information in each frame. Still, stretched over four frames, a motion gesture is evolving, that describes the two clusters of points corresponding to the arms to close in. Specifically, the spatial relation between points in each individual frame is less expressive to infer a gesture than the temporal dependencies of points across consecutive frames.

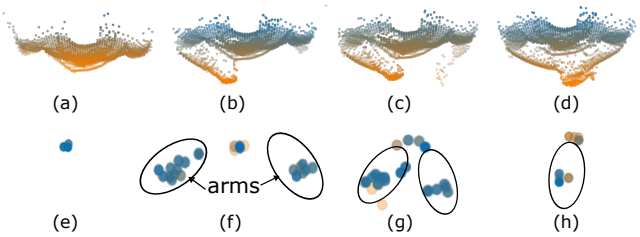


Fig. 3. (a), (b), (c), and (d) are point clouds of four frames from a single gesture of the UBPG dataset which is essentially a closing in of the two arms from a wider position. A similar gesture captured by the mmWave radar is shown in (e), (f), (g), and (h) over four frames.

4 Proposed Model

In this section, we describe Tesla, an MPNN based graph convolution approach tailored for inferring gestures from motion point clouds.

The architecture of Tesla is depicted in Fig. 4. First, in order to make the prediction model robust against possible spatial transformations of input gestures (e.g., rotation, translation, scaling, etc.), we apply a TFNet [58] module on the input point cloud. This trainable module is responsible for producing a dynamic transformation for each input gesture's entire feature map to transform the possibly skewed points to a rigid, uniform, and canonical point cloud, which in turn makes the recognition in the following layers simple. Next, we apply our proposed TeslaConv layer on the output of TFNet, which includes two steps: *Graph Generation* and *Graph Processing*. In the Graph Generation phase, a temporal graph is created from motion point clouds through the proposed Temporal Graph K-NN algorithm, which connects each point to its nearest neighbors from previous frames to reflect the temporal pattern of gesture. In the Graph Processing step, we apply the proposed MPNN scheme that learns the representation of each point according to the structure of the generated graph. Additionally, we optimize this layer by integrating an self attention mechanism in the message passing scheme to improve the performance of the graph processing. Furthermore, it decreases the computational complexity of the model by eliminating the need for removing outliers of the dataset explicitly. In the following we will present more details about each step of the TeslaConv.

4.1 Graph Generation

Consider a point cloud $X = \{x_1, ..., x_n\} \subseteq \mathbb{R}^F$ where each point is represented by a feature set of $x_i = \{f_i^1, ..., f_i^F\}$. In motion point clouds the frame number f_i^s of each point is also a dimension of the feature set, i.e., $f_i^s \in x_i$. The K-NN graph $\mathcal{G} = \{X, \mathcal{E}\}$ is obtained through the Graph K-NN algorithm where $\mathcal{E} \subseteq X \times X$ is the set of directed edges between each point and its closest neighbours in the Euclidean space.

As illustrated in Fig. 5, in the graph generation phase, for each point, we use Temporal Graph K-NN to find the nearest neighbors only from the previous frames. For swipe-left gesture in Pantomime dataset, the comparison between Graph K-NN and Temporal Graph K-NN in the graph structure is shown in Fig. 6. The trend in the direction of

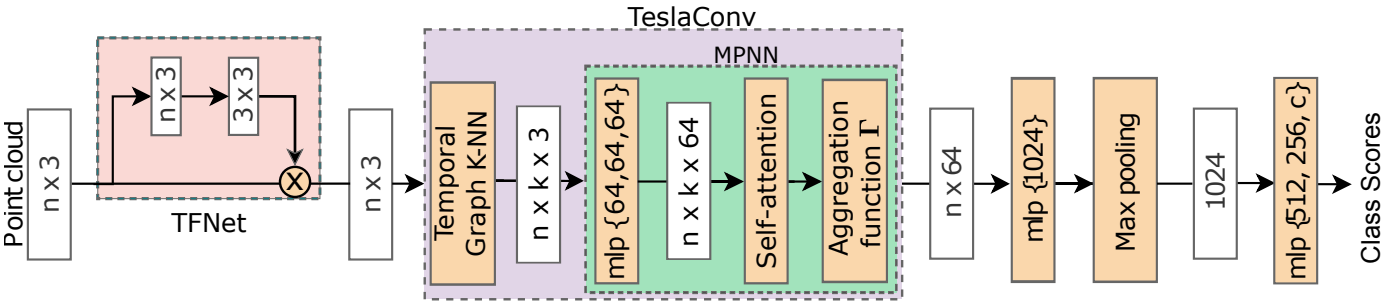


Fig. 4. The architecture of Tesla- Having multiplied the input point cloud by a 3×3 spatial transformation matrix of TFNet, the transformed output is fed into TeslaConv. In TeslaConv a temporal graph is created using Temporal Graph K-NN module and the proposed message passing scheme (section 4.2) is applied. Afterwards, to represent the gesture as a fixed-sized vector, an MLP of size 1024 followed by a max pooling is performed. Finally, a three layered MLP with respective sizes of 512, 256, and c (the number of classes) is used to predict the class scores of the gesture.

the arrows in Fig. 6.(c) shows the temporal evolution of the gesture whereas that of Fig. 6.(b) is irrelevant to the temporal pattern.

In the first step of Temporal Graph K-NN, we normalize the feature set of each input point using batch-wise min-max normalization.

$$x_i = \frac{x_i - min(x)}{max(x) - min(x)}. \tag{1}$$

in which, $min(x)$ and $max(x)$ are the minimum and maximum values of each dimension of x over a batch of input gestures, respectively. In the second step, we multiply the temporal dimension of x_i (f_i^s) by a hyperparameter α to control the trade-off between temporal and spatial features. Setting α to a large number (e.g., 100) forces the model to find the nearest neighbors only from the previous frame, while small numbers of α (e.g., 0) gives the model more freedom in choosing the nearest neighbors from the whole non-masked set.

$$f_i^s = \alpha f_i^s. \tag{2}$$

To find the nearest neighbors only from previous frames, we introduce a masking scheme. The masked set of points $\mathcal{F}_{x_i}$ for x_i is obtained through:

$$\mathcal{F}_{x_i} = \{x_j : x_j \in X \wedge f_j^s > f_i^s\} \tag{3}$$

Furthermore, the distance between two points is defined as the Euclidean distance of all the corresponding features of points including f_i^s and is calculated according to:

$$D_{x_i,x_j} = \begin{cases} ||x_i - x_j|| : x_i, x_j \in X, & \text{if } x_j \notin \mathcal{F}_{x_i}, \\ \infty, & \text{otherwise}, \end{cases} \tag{4}$$

where D_{x_i,x_j} denotes the distance between x_i and x_j and $||.||$ is Euclidean norm operator. Finally, the introduced masked distance function is used to find the nearest neighbors in Temporal Graph K-NN.

4.2 Graph Processing

As shown in Fig. 5, in the graph processing phase, the representation of each point is calculated through the proposed MPNN layer based on the temporal graph. In each layer, the hidden representation of each point is updated through an aggregation function on the point features except for f_i^s from the previous layer and the messages of its neighbours according to:

$$\begin{aligned} h_i^0 &= x_i \setminus \{f_i^s\}, \\ h_i^l &= \underset{j:(i,j)\in\mathcal{E}}{\Gamma}(M_\theta(h_i^{l-1}, h_j^{l-1})), \end{aligned} \tag{5}$$

in which, h_i^l is the hidden representation of point i in MPNN layer l, $\setminus$ is the set subtraction operator, message function $M_\theta : \mathbb{R}^F \times \mathbb{R}^F \to \mathbb{R}^{F'}$ is a non-linear function with a set of trainable parameters θ and is usually implemented using MLP architectures, Γ is a channel-wise symmetric aggregation function (e.g. Σ, max, or mean) applied on the messages of the edge emanating from each neighbor.

The choice of M and Γ significantly affects the properties and the performance of the model in Eq. (5). For example, setting $M_\theta(h_i, h_j) = M_\theta(h_i)$ causes the model to only capture the global features of point clouds without considering the local structures. On the other hand, setting $M_\theta(h_i, h_j) = \bar{M}_\theta(h_i, h_j - h_i)$, provides information about the local relations of the neighbouring points. In this paper, we use the second setting of message function to help capture the local dependencies as well as the global structure.

To decrease the effect of noisy points, we integrate a *scaled-dot multi-head self-attention* mechanism [59] shown in Fig. 7 into the message function. The goal is to let the incident edges to point i decide their relative importance in determining the updated representation of the point. Let $\mathcal{M}_i = \bigoplus_{j:(i,j)\in\mathcal{E}} \bar{M}_\theta(h_i, h_j - h_i), \mathcal{M}_i \in \mathbb{R}^{k \times F'}$ denote the array representation of the set of the messages of incident edges for each point i and $\bigoplus$ is the concatenation of messages along the first dimension. A set of query Q_b^i, key K_b^i, and value V_b^i for point i in a single-head self attention is calculated through:

$$Q_b^i = \mathcal{M}_i W_b^Q, \quad K_b^i = \mathcal{M}_i W_b^K, \quad V_b^i = \mathcal{M}_i W_b^V \tag{6}$$

where, $W_b^Q \in \mathbb{R}^{F' \times d_k}, W_b^K \in \mathbb{R}^{F' \times d_k}, W_b^V \in \mathbb{R}^{F' \times d_v}$ are learned linear projections to d_k, d_k and d_v dimensions, respectively and b is the head index. Then, the single-head

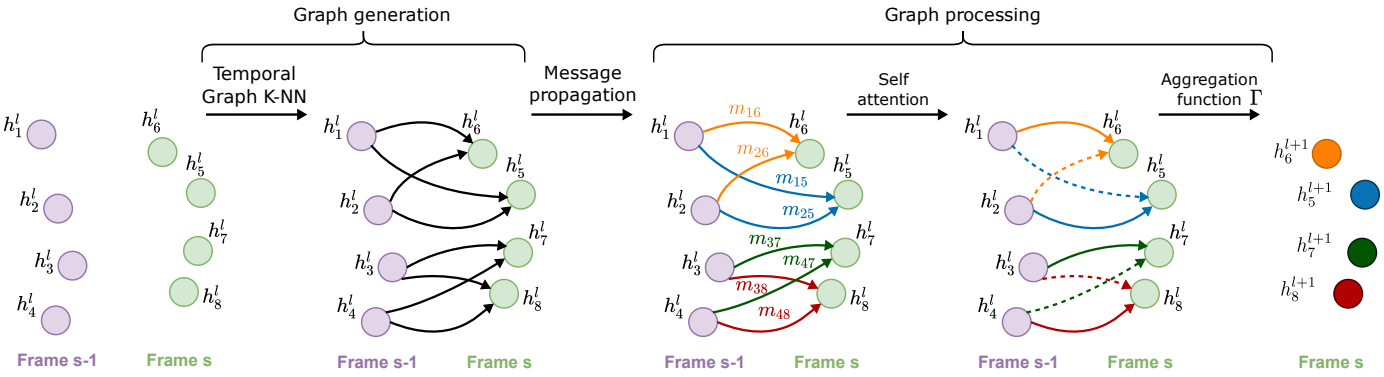
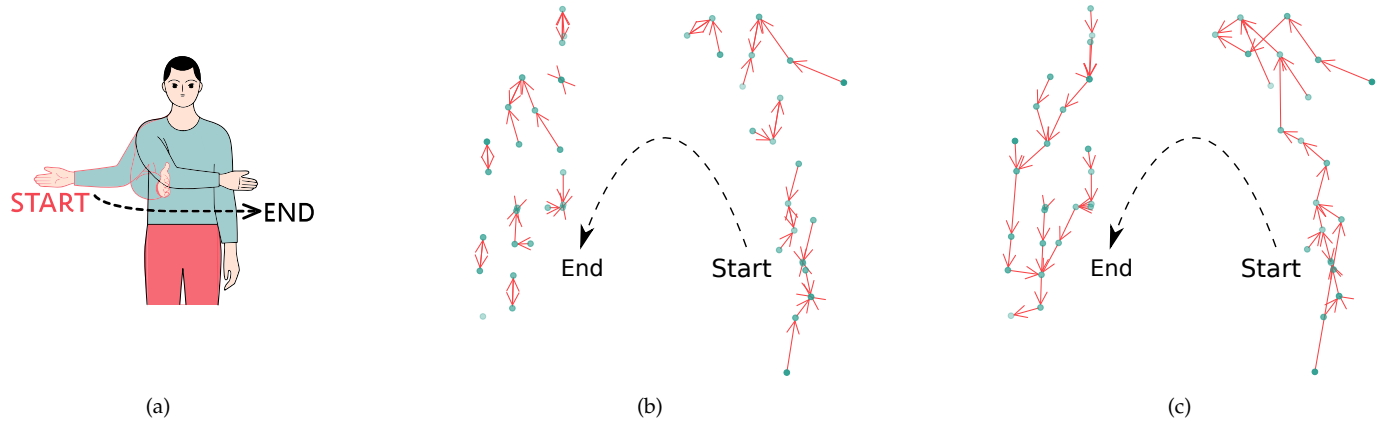


Fig. 5. TeslaConv layer- For points in each frame, a directed edge is connected from the nearest neighbours in the previous frames through Temporal Graph K-NN. Next, messages are propagated according to the direction of the edges and a multi-head self-attention is performed on them. Finally, the representation of each point is obtained by Γ aggregation function on incoming messages.



(a) (b) (c)

Fig. 6. Intuition behind Temporal Graph K-NN- (a) The schematic of the swipe-left gesture from the Pantomime dataset (b) Generated graph using Graph K-NN (c) Generated graph using Temporal Graph K-NN with $\alpha = 100$. Both point clouds are shown from a top view and K is equal to 1 for simplicity.

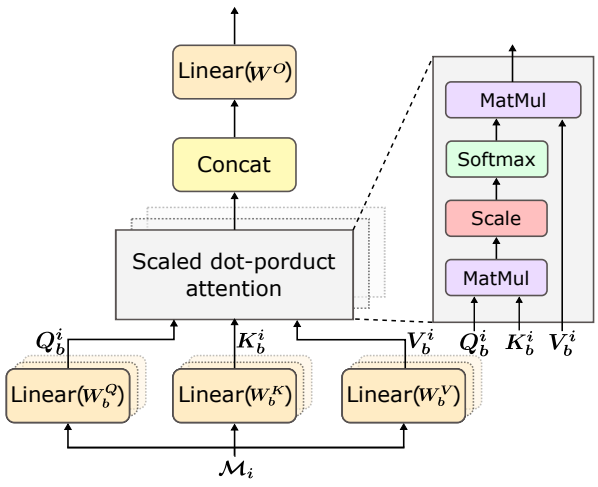


Fig. 7. Multi-head Self Attention mechanism- **Linear** refers to multiplication with corresponding learnable weights (Eq. (6) & Eq. (8)), **Scaled dot-product attention** is formulated in Eq. (7), and finally, **Concat** is the concatenation operation in Eq. (8).

self attention on the messages of each point is calculated through:

$$H_b^i(Q_b^i, K_b^i, V_b^i) = softmax\left(\frac{Q_b^i \times (K_b^i)^T}{\sqrt{|K_b^i|}}\right) \times V_b^i. \quad (7)$$

in which, $\times$ is matrix multiplication operator. Moreover, employing the multi-head approach allows the model to calculate the attention scores using different sub-spaces at the incident edges' messages as well as a more stable learning process. In this work we employ $m = 8$ parallel attention layers with f_o/m dimensions, where f_o is the number of dimensions of incident messages after performing message function. The final multi-head output is obtained by

$$A(\mathcal{M}_i) = (\mathop{\Big\|}_{b=1}^{m} H_b(Q_b^i, K_b^i, V_b^i))W^O, \quad (8)$$

Where $W^O \in \mathbb{R}^{md_v \times F'}$ are trainable weights and $\|$ is the concatenation operator. Thus, the message passing part of the TeslaConv layer (Eq. (5)) can be updated as:

$$h_i^l = \mathop{\Gamma}_{j:(i,j)\in\mathcal{E}} A(\mathcal{M}_i^{l-1}) \quad (9)$$

For results on the effect of self-attention mechanism see section 6.3.

4.3 Permutation Invariance

Since the permutation of points in X does not alter the nature of the gesture, the prediction model should be permutation invariant with respect to the order of the input points. This can be proved in two steps for our approach.

First, Temporal Graph K-NN, introduced in section 4.1, uses symmetric aggregations (min and max) and calculates the Euclidean distance between points which leads to a permutation invariant graph generation process.

Second, in this work we use max as the aggregation function in Eq. (9):

$$h_i^l = \underset{j:(i,j)\in\mathcal{E}}{max} A(\mathcal{M}_i^{l-1}) \tag{10}$$

Since max in Eq. (10) and the global max-pooling function shown in Fig. 4 are symmetric functions, the output of the layer is permutation invariant w.r.t the input.

5 Implementation

In this section, we present the implementation details of Tesla-Rapture in terms of preprocessing pipeline, training and inference phases, and real-time gesture recognition interface on Raspberry PI 4.

5.1 Preprocessing

To prepare the data for Tesla model and study the effect of different hyperparameters, we design a preprocessing pipeline. Angle and distance normalization, frame division, and point re-sampling are the steps of the pipeline in the mentioned order.

5.1.1 Angle and distance normalization

To reduce the effect of angle and distance of the participant w.r.t. the antenna center-line on the accuracy, data normalization is performed. To do so, we use affine-geometric transformation matrices to rotate and translate the data to the reference point (1.5m distance and 0 angle in Pantomime and 1m distance and 0 angle in RadHAR).

5.1.2 Frame divider

To study the effect of number of frames on system accuracy, complying with the temporal order of points, we distribute them in different number of frames (2, 4, 8, 16, 32, 64). Assume S is the desired number of frames and n is the total number of points ordered based on the time at which each point was received. We consider first n/S points as the first frame, second n/S points as the second frame and so on. Although we re-define the frame structure, the temporal order of points is still preserved.

5.1.3 Point re-sampling

To study the effect of number of points in each frame on the system performance, we employ a density-based re-sampling strategy introduced by Cohen et al. [60] to preserve the spatial structure while fixing the number of points in each frame. Considering n/S as the desired number of points in each frame, to reduce the number of points we use $\mathcal{K}$-means algorithm and set $\mathcal{K}$ equal to n/S and select the centroids of the clusters as the points in the frame. To increase the number of points in the frame to n/S, we iteratively apply Agglomerative Hierarchical Clustering (AHC) and add the centroids of the clusters as new points to the frame until we have the desired number of points.

5.2 Data Augmentation

Different data augmentation techniques are applied to improve the generalizability of the system in terms of different angles, distances, and scales. We apply the following augmentations to each batch during the training phase for all the models:

- Random translation up to 10cm
- Random scaling between 0.8 to 1.25
- Random point-wise translation (jitter) based on a Gaussian distribution with $\mu = 0$ and $\sigma = 0.01$
- Random clipping of 0.03m
- Random shuffling of the point cloud representation preserving the spatial and temporal features

5.3 Training and Inference

The infrastructure used for training and inference phases has 64GB of RAM and is equipped with a Tesla V100 16GB GPU. The model is implemented using PyTorch [61] and PyTorch Geometric [62]. We utilize early stopping mechanism in the training phase with a patience of 100 epochs. To do so, if no improvement on validation set accuracy is observed within the patience period, training is stopped and the best model is saved. The loss function used for training the model is cross-entropy between class scores and one-hot encoded labels. To minimize this loss function, we use Adam Optimizer [63] with a step-decay strategy to decrease learning rate:

$$L_r = L_{init} \cdot d_r^{\lfloor \frac{e}{e_r} \rfloor} \tag{11}$$

where L_r is the learning rate used at each epoch, L_{init} is the initial value of the learning rate, d_r is the drop rate after every e_r epochs, e is the current epoch and $\lfloor \cdot \rfloor$ is the floor operator. In our setup L_{init} is 0.001, d_r is 0.5, and e_r is 20.

5.4 Real-time Implementation

We implement Tesla-Rapture for real-time gesture recognition on Raspberry PI 4 device with 8GB RAM, as an example embedded device with constrained computing resources.

For recognizing gestures in real-time, we develop an algorithm which uses Tesla model as classifier. We categorize each captured frame into two sets of *active frame* and *idle frame*. *Idle frames* are frames in which no notable movement is observed and the rest are considered as *active frame*. In Algorithm 1, we use a set of consecutive idle frames as a delimiter for different gestures. A similar approach is employed in different gesture recognition systems like DoubleFlip [64] and WristRotate [65] or even in voice assistants, e.g., [66]. Gesture recognition is performed whenever a minimum number of active frames are identified. Thresholds for minimum active frames, gesture delimiter, and maximum number of points for idle frames are denoted as min_frames, $idle\_frame\_delimiter$, $idle\_frame\_threshold$, respectively and tuned empirically.

Algorithm 1: Real-time recognition algorithm

Result: Recognized gesture
$frame\_list = []$, $min\_frames = 2$,
$idle\_frame\_count = 0$,
$idle\_frame\_delimiter = 10$,
$idle\_frame\_threshold = 3$;
while *Receive* $frame\_data$ *from Radar* **do**
  if *len(*$frame\_list$*)* $\geq min\_frames$ *and*
   $idle\_frame\_count \geq idle\_frame\_delimiter$
   then
    preprocess($frame\_list$);
    perform_recognition($frame\_list$);
  end
  if *len(*$frame\_data$*)* $\leq idle\_frame\_threshold$
   then
    $idle\_frame\_count+ = 1$;
    $continue$;
  end
  $idle\_frame\_count = 0$;
  append $frame\_data$ to $frame\_list$;
end

	Pantomime	RadHAR	mHomeGes
Participants	41	2	25
Number of classes	21	5	10
Max. range (m)	5 m	1.5 m	3 m
Environments	5	1	7
Frame rate (fps)	30	60	10
Training samples	7000	12097	22,000

TABLE 1
Comparison of the three datasets.

The real-time recognition algorithm is implemented on a Raspberry PI 4 with a connected IWR1443 Radar responsible for sensing the human movement (see section 3.1). A Cortex-R4F built-in micro-controller is employed in the radar and the universal asynchronous receiver-transmitter protocol realizes data transfer. We configure the device to capture frames at a rate of 30 fps with a range resolution of 0.047 m, a velocity resolution of 0.87 m/s, and a maximum velocity of 6.9 m/s up to a maximum range of 5 m. The starting frequency is 77GHz and our selected range resolution dictates a bandwidth of 3.19GHz.

6 Evaluation

This section presents the performance evaluation of the Tesla model and Tesla-Rapture system in terms of recognition accuracy and time complexity.

6.1 Datasets

For evaluation purpose, we use three radar generated point cloud datasets: Pantomime [16], RadHAR [43], and mHomeGes [50]. The comparison between three datasets is shown in Table 1. All datasets were acquired using a 77 GHz IWR1443 millimeter wave radar. The gestures in Pantomime are divided into three sets: *Easy* (9 classes), *Complex* (12 classes) and *All* (21 classes) based on the execution difficulty. The *Easy* set comprises single-hand gestures that are easy to perform and remember. The *Complex* set comprises bimanual, linear, and circular gestures. Finally, *All* consists of gestures from both sets. The training data in RadHAR is collected from one anchor position of 1.5m, whereas the training data in Pantomime and mHomeGes are collected from 4 and 13 anchor positions between 1.5 to 5m and 1.2m to 3m, respectively. For evaluating the model on Pantomime and RadHAR datasets, we employ the same train, validation, test splits provided by Pantomime and RadHAR authors, respectively. For mHomeGes, we use the same 20%/20%/60% test/validation/train split for all the models.

6.2 Evaluation Metrics

The performance of models are evaluated using recognition accuracy (Acc.), Area Under ROC Curve (AUC), which quantifies the discriminatory power of the classifier, and Average Precision (AP), which summarizes a precision-recall curve as the weighted mean of precisions achieved at each threshold, with the increase in recall from the previous threshold used as the weight:

$$AP = \sum_{th}(R_{tr} - R_{tr-1})P_{tr} \tag{12}$$

where P_{tr} and R_{tr} are the precision and recall at threshold tr. We also report the confusion matrix to analyze performance further and inspect conflicting gestures.

6.3 Hyperparameter Tuning

The results of hyperparameter tuning of the model on Pantomime validation dataset are illustrated in Fig. 9. In order to tune each point's neighbors number (k) and the value of α in Temporal Graph K-NN and the number of TeslaConv layers, different combinations of parameters are used to train the model and test it on the validation set. According to the cross-validation process, best results are obtained using one layer of TeslaConv with $k = 32$ and $\alpha = 10$. Increasing the complexity of the model by adding more layers does not contribute to the accuracy of the model. Although, increasing k leads to a more complex model since Temporal Graph K-NN generates denser graphs (see Fig. 4), in most of the cases the accuracy is enhanced as demonstrated in Fig. 9. In general, no clear trend is observable when it comes to changing α indicating that performance of different α values is not independent from values of k and the number of layers.

We choose two sets of hyperparameters: Tesla model with $k = 32$ and $\alpha = 10$, the best performing one in terms of accuracy, and Tesla-V (Tesla-Vanilla) model with $k = 2$ and $\alpha = 10$, reasonably accurate but faster than Tesla in terms of prediction time.

In Fig. 10 the impact of the number of frames and the number of points in each frame on the accuracy is evaluated on three different datasets. Six settings of different combinations of the number of frames and the number of points per frame are considered while keeping the total number of points (=number of frames $\times$ number of points per frame) in each gesture constant (1024). Increasing the number of frames up to 32 for Pantomime and mHomeGes datasets, improves the accuracy. However, adding more frames than

32 to the gesture in these two datasets decreases the accuracy. For RadHAR dataset, the optimal number of frames is 8. These observations indicate that both number of frames and number of points in each frame play important roles in performance of the system.

Additionally, to illustrate the effect of self-attention mechanism on the performance of the model, we train the Tesla without the self-attention mechanism on the training set of the Pantomime dataset. The overall accuracy of the trained Tesla without self-attention on the validation set is **95.2%** (**3%** drop compared to the model with self-attention) indicating the positive effect of self-attention in improving the performance of the model. Moreover, we train Tesla with number of heads, m, equal to 1 and the accuracy is **97.7** which is **0.4%** behind the model with 8 heads showing that using multiple heads improves the performance of the model.

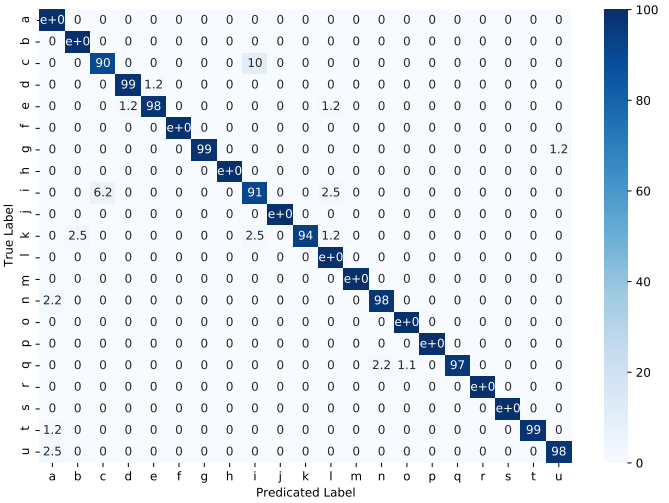


Fig. 8. Confusion matrix of Tesla on Pantomime dataset

6.4 Classification Results

6.4.1 Overall Results on Pantomime dataset

In Table 2, the performance of Tesla and Tesla-V on Pantomime dataset is compared to baseline models of *PointNet* [14], *PointNet++* [51], *O&H* [55], *PointGest* [17], *RadHAR* [43], *PointLSTM* [15], *Pantomime* [16], P4T [67], and *DEC* [22]. In PointNet, PointNet++, and DEC, the frames are aggregated through time dimension into a single frame representing the whole gesture, since they are designed to classify static point clouds. While the rest of the models aim to classify motion point clouds. Moreover, from input data representation perspective, O&H and RadHAR work on voxels whereas the rest of them directly operate on point clouds. In case of Pantomime and PointGest, gestures are represented with 8 frames (same number as in the original papers) since they are computationally demanding and not feasible to run with more frames on the same infrastructure. As illustrated in Table 2, our Tesla model outperforms all baselines in every category, in terms of accuracy, AUC, and AP. Additionally, Tesla model increases the accuracy of state of the art by **0.9**%, **4.2**%, **3.1**% in Easy, Complex, and All settings, respectively, as well as achieving **100**% AUC in both Complex and All. Furthermore, Tesla-V model performs rather efficiently compared to baselines and Tesla, ranking 2nd on Complex and All and 3rd (only 0.1% behind 2nd) on Easy when it comes to accuracy.

6.4.2 Different Environments

We also evaluated Tesla model with different environments on Pantomime dataset, comparing to the closest competitor. Following the same approach as [16], the model is trained on data acquired in Open and Office settings and tested against five different environments reported in Table 3. We manage to improve accuracy up-to 10% in all environments except for Open. This arises from the fact that the frames in cluttered environments like Through-wall are sparser compared to less cluttered environments , e.g., Open. Therefore, the spatial distribution of the frames in the train set is different from that of the test set. Consequently, the models capturing spatial features and fusing them through Long Short-Term Memory (LSTM) layers i.e. Pantomime, fail to generalize well (see section 3.2). On the contrary, Tesla model, recognizes gestures based on their temporal structures which leads to a more robust prediction in unseen environments.

6.4.3 Different Speeds

In addition, the effect of gesture speed is illustrated in Table 3. The models are trained on gestures performed with Normal speed and tested on Slow, Normal, and Fast speeds. Tesla model outperforms Pantomime in Normal and Fast articulation speeds. However, in setting *Slow*, we are behind state of the art.

6.4.4 Different Distances and Angles

For measuring the robustness of the prediction against the position of the participant w.r.t. radar, we compared the performance of Tesla, on different angles and distances. As shown in Fig. 11, our Tesla model outperforms Pantomime in every setting of angle and distance in terms of both accuracy and AUC. When it comes to extreme setups i.e. 5m distance, $-45°$ and $45°$ angels, Tesla is significantly ahead of Pantomime improving the accuracy up to 21%. Furthermore, with the increase of distance, the performance drop in our Tesla is less than 10%, whereas Pantomime degrades in accuracy with an exponential rate (almost 30%). Given the change in the distribution of point clouds in different configurations of the radar (see section 3.2), Pantomime fails to generalize since it extracts spatial features from each frame, fusing them to identify temporal pattern. However, Tesla recognizes gestures based on the temporal graph which is more robust to angle and distance.

6.4.5 Conflicting gestures

The confusion matrix of Tesla on the *ALL* set of Pantomime dataset is shown in Fig 8. There are a few minor conflicts in the gestures, but the main source of confusion is between gestures *'i': throw* and *'c': lift* which are shown in Fig. 12. The first reason is that the gestures are similar since both of them are performed in mainly z-axis. The second reason is that the radar has a lower resolution in elevation angle given the less number of antennas in that direction [16].

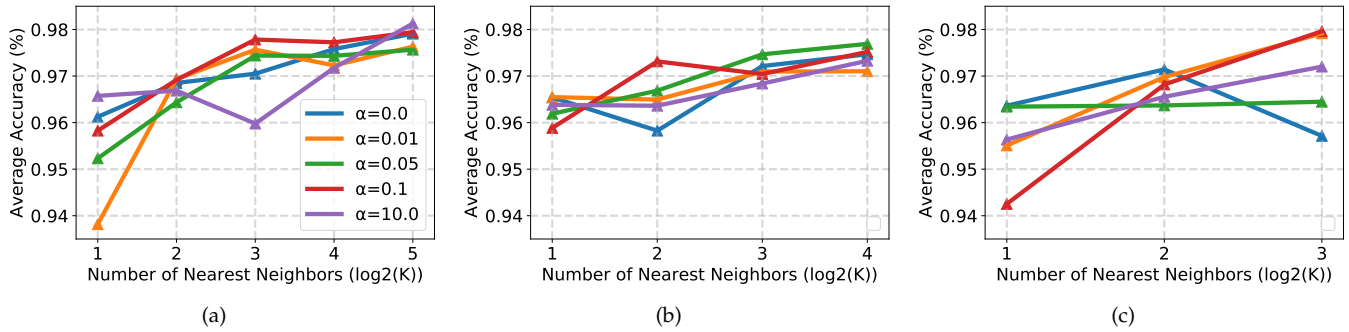


Fig. 9. The effect of hyper parameters. (a) One, (b) two and (c) three layers.

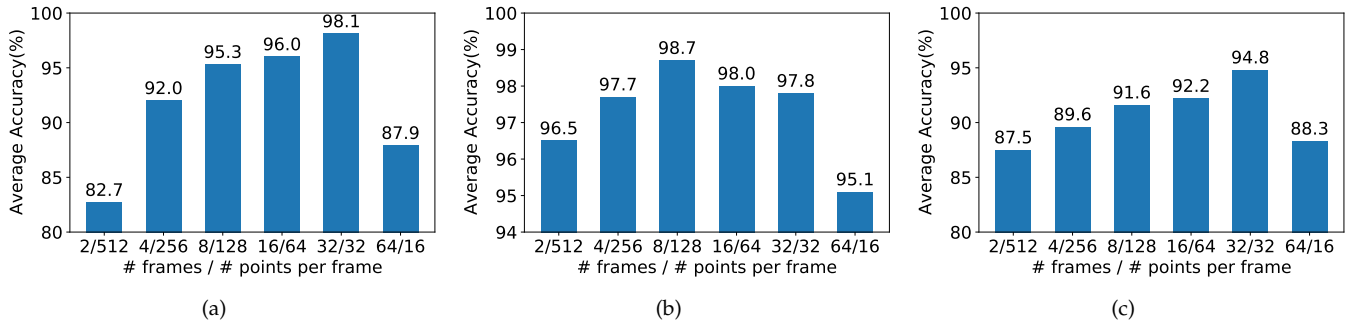


Fig. 10. The impact of the # of frames and points per frame on avg. accuracy. for (a) Pantomime dataset (b) RadHAR dataset and (c) mHomeGes dataset

	Easy			Complex			All		
Model	**Acc.**	**AUC**	**AP**	**Acc.**	**AUC**	**AP**	**Acc.**	**AUC**	**AP**
PointNet	79.7	98.4	81.1	82.5	98.7	82.6	81.6	99.4	82.2
PointNet++	79.7	98.1	80.5	84.9	99.0	85.5	83.6	99.4	84.2
O&H	77.7	96.0	79.7	83.2	98.1	84.1	79.1	97.9	80.2
PointGest	83.3	98.5	85.6	88.4	99.5	89.2	86.3	99.4	88.8
RadHAR	91.6	98.9	91.7	94.3	99.6	94.5	89.9	99.5	90.3
PointLSTM	85.1	99.1	86.3	92.1	99.8	93.6	90.7	99.7	91.8
P4T	78.9	97.0	79.4	79.8	97.0	80.2	78.2	98.1	79.5
Pantomime	96.6	**99.8**	96.6	95.1	99.8	95.4	95.0	99.9	95.3
DEC	81.9	98.2	82.4	89.1	99.3	90.2	86.0	99.4	86.4
Tesla-V	96.2	99.7	96.8	99.1	100	**99.1**	96.6	99.9	96.7
Tesla	**97.5**	**99.8**	**97.6**	**99.3**	**100**	**99.1**	**98.1**	**100**	**98.2**

TABLE 2
Comparison with the state of the art on the Pantomime dataset. Acc., AUC, and AP are reported in percentages. The best results per column are denoted in bold typeface.

6.4.6 *Overall results on RadHAR dataset*

In Table 4, the results of different models on RadHAR dataset are illustrated. SVM, MLP, *Bi-directional LSTM*, and *RadHAR* use voxels as input. As shown in Table 4, Tesla model outperforms baselines in measures of accuracy, AUC, and AP. Moreover, Tesla-V ranks third in the table in terms of accuracy, AUC, and AP.

6.4.7 *Overall results on mHomeGes dataset*

The results of different models on mHomeGes dataset are shown in Table 5. Tesla model outperforms baselines in accuracy, AUC, and AP. The margin between Tesla and the closest competitor, P4T, is 9.09%. Moreover, Tesla-V ranks second in the table in terms of both accuracy and AUC.

6.5 Time Complexity Results

In Fig. 13, time complexity comparison between Tesla and Tesla-V and baselines on Pantomime dataset on a Tesla V100 Graphical Processing Unit (GPU) with 16GB of memory is presented. To evaluate the efficiency of the model, we measure four different metrics of *average inference time*, *GFLOPs*, *number of trainable parameters*, and *size* of the trained model. For measuring inference time, two settings of batch size 1 and 16 were considered and the average time of 10 forward passes were gathered after warming up the infrastructure by running a few batches. Each category of measurements in Fig. 13 are scaled based on the maximum value of the category. According to Fig. 13, Tesla-V model has the lowest aggregate complexity among all the models. Furthermore, Tesla model, which is the best performing one in terms of

Setting	Pantomime			Proposed		
	Acc.	AUC	AP	Acc.	AUC	AP
Factory	89.11	99.79	91.03	**97.14**	**99.96**	**97.63**
Restaurant	81.13	**98.84**	84.01	**82.14**	98.19	**88.40**
Office	93.40	99.86	93.65	**97.14**	**99.94**	**97.24**
Open	**96.12**	**99.94**	**97.11**	94.36	99.88	96.21
Through-wall	64.43	97.24	68.72	**74.64**	**98.51**	**81.56**
Slow	**85.00**	**99.33**	**86.62**	76.19	98.69	86.19
Normal	94.05	99.90	94.27	**95.95**	**99.95**	**96.53**
Fast	92.14	99.68	92.93	**94.28**	**99.87**	**94.62**

TABLE 3
Comparison with Pantomime model (the closest competitor) on different settings of the Pantomime dataset. The best Acc., AUC, and AP per row are denoted in bold typeface.

Model	Acc.	AUC	AP
Support Vector Machine (SVM)	63.74	-	-
MLP	80.34	-	-
Bi-directional LSTM	88.42	-	-
RadHAR	90.47	-	-
PointLSTM	94.11	98.70	94.74
P4T	67.82	91.33	68.64
Pantomime	94.19	99.65	94.95
DEC	96.24	99.62	96.30
Tesla-V (ours)	95.49	99.48	95.59
Tesla (ours)	**98.71**	**99.91**	**98.92**

TABLE 4
Comparison with the state of the art on the RadHAR. The Accuracy is reported in percentages. The performance of *SVM*, *MLP*, *Bi-directional LSTM*, and *RadHAR* are reported from [43]. The best results per column are denoted in bold typeface.

accuracy, ranks 3rd in total, just behind PointNet, a model that does not take into account the temporal dependency, therefore, having a much lower accuracy. Compared to the most accurate competitor (Pantomime), Tesla-V is 18 and 8 times faster in inference with batch sizes 16 and 1 respectively; and 40 times computationally efficient in terms of GFLOPs. In addition, computationally closest competitor is PointNet which has almost the same inference time and

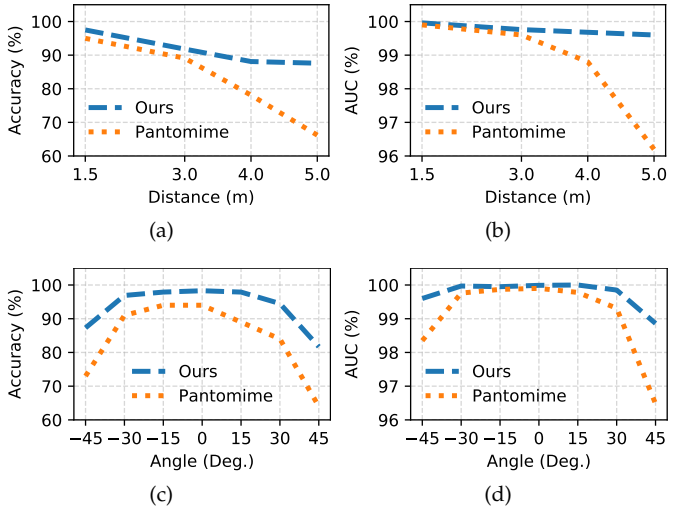


Fig. 11. Comparison between Pantomime and Tesla models. (a), (b) on different distances and (c), (d) on angles in terms of accuracy and AUC

Model	Acc.	AUC	AP
PointNet	80.45	99.05	75.22
PointNet++	79.77	98.97	75.22
PointGest	68.36	96.52	65.71
PointLSTM	83.70	97.93	83.82
P4T	85.71	98.66	85.75
Pantomime	80.24	99.12	75.86
DEC	78.99	97.56	79.08
Tesla-V (ours)	90.35	99.34	90.92
Tesla (ours)	**94.80**	**99.53**	**95.87**

TABLE 5
Comparison with the state of the art on the mHomeGes dataset. The best results per column are denoted in bold typeface.

Model	Size	IT-1	IT-16	FLOPS	Params
PointNet	43.00	6.39	**0.64**	0.45	3.47
PointNet++	18.40	7.03	1.08	1.68	1.47
PointGest	24.80	37.96	7.44	13.47	1.66
PointLSTM	**5.28**	22.00	8.00	4.03	**1.24**
Pantomime	45.80	43.00	12.46	15.16	3.26
P4T	76.20	25.71	1.20	4.87	19.96
DEC	6.80	72.60	6.47	0.87	1.69
Tesla-V (ours)	6.30	**5.23**	0.66	**0.40**	1.56
Tesla (ours)	6.30	13.04	4.30	1.29	1.56

TABLE 6
Comparison to the state of the art in terms of computational complexity

GFLOPs while falling behind Tesla-V by 16.5% when it comes to recognition accuracy.

Moreover, we have the explicit metrics for each model in Table 6. Tesla-V in terms of inference time with batch size equal to 1 and FLOPS is better than the rest. Although in terms of model size, inference time with batch size equal to 16, and number of parameters Tesla-V ranks either second or third, when it comes to the normalized aggregated metric (Fig. 13, it is ahead of the rest of the models, making it suitable for real-time gesture recognition on embedded devices.

6.6 Real-time Implementation Evaluation

In this part we evaluate inference time and performance of the model on a Raspberry PI 4 device with 8GB of RAM. The setup for the real-time testbed is shown in Fig. 15.

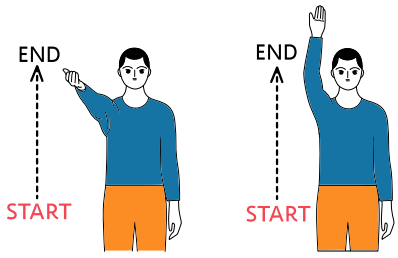


Fig. 12. The conflicting gestures from Pantomime dataset: (a) Gesture 'i': throw, (b) Gesture 'c': lift

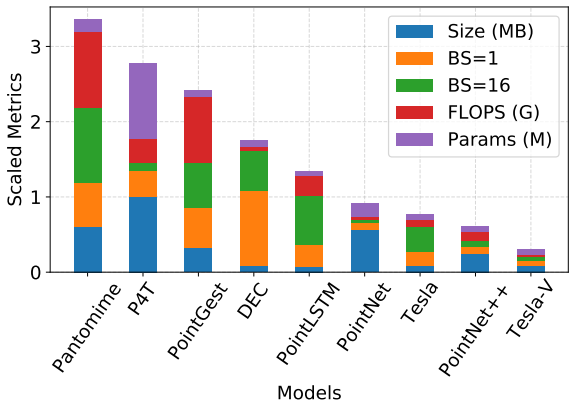


Fig. 13. Scaled model complexity comparison. Each metric is scaled between 0 and 1. Size: model size, BS: average inference time a batch size, Params: trainable parameters

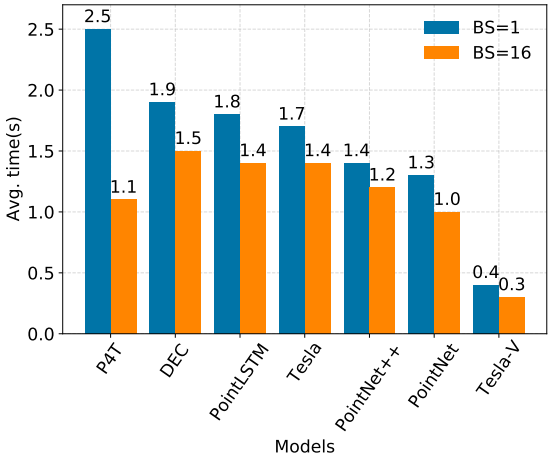


Fig. 14. Average inference time per gesture of proposed models on a Raspberry PI device

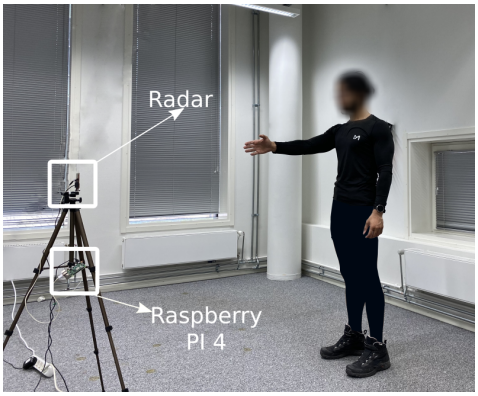


Fig. 15. Tesla-Rapture setup for real-time evaluation using an IWR1443 radar and Raspberry PI 4 device

6.6.1 Inference time

In Fig. 14, the average inference time on a Raspberry PI 4 with 8GB of memory with batch sizes of 1 and 16, for DEC, PointLSTM, Tesla, PointNet++, PointNet, and Tesla-V are illustrated. Since, Pantomime and PointGest require more than 8GB of RAM, their implementation on Raspberry PI is not feasible. Among the implemented models, Tesla-V is able to predict gestures in 0.4s and 0.3s with batch sizes 1 and 16 respectively, making it the only model predicting a gesture in less than half a second. As a result, Tesla-Vis the only suitable model for integration into the real-time gesture recognition interface Tesla-Rapture, since inference time of more than one second in the case of all other models, is not fast enough for real-time user experience.

6.6.2 Performance

To evaluate the performance of the purposed gesture recognition system, Tesla-Rapture, the pipeline shown in Fig. 1 is implemented. The prediction model in this system is Tesla-V (the only model with inference time of less than one second) and the inference is done on a Raspberry PI device connected to an IWR1443 radar for gathering gestures.

Since Pantomime dataset does not have a class for rejecting gestures (*no-gesture* class), 2 hours of moving point clouds in which there were random movements of participants as well as idle frames were recorded. The training data from Pantomime dataset was combined with *no-gesture* samples to train the model. As a result, to train Tesla-V for the real-time system, we used 22 classes including a *no-gesture* class to reject the gestures that do not belong to the gesture set of Pantomime dataset.

In the evaluation phase of the Tesla-Rapture, we asked 5 participants to perform each class of gesture for 10 times as well as doing random movements in front of the radar (like walking, staying idle, and doing some random gestures other than the original gesture set). To do so, we showed gesture videos to participants and asked them to perform each gesture a few times before the actual evaluation. In the evaluation round, we showed the name and the schematic view of the gesture in a random order on the screen and the participant performed the gesture. As shown in algorithm 1, we use a few idle frames as a gesture delimiter. Consequently, between each gesture there was one second gap. The overall accuracy of the real-time system is **90.53%** and the false-alarm rate is **4.4%**.

7 Discussion

Gesture Recognition. Introducing Tesla-Rapture system, as a fast and accurate gesture recognition interface is a step forward in human-computer interaction scenarios for integration with many off-the-shelf devices. Given the robustness of the system in different environments, angels, and distances as well as real-time performance, Tesla-Rapture system can be incorporated into a wide range of applications e.g., smart-homes, vehicular settings, and human-robot interaction. Furthermore, the model can be trained on a customized set of gestures and deployed on Tesla-Rapture for a specific real-time application.

Speed vs. Accuracy. High performance of Tesla makes it suitable for sensitive applications in which the accuracy cannot be compromised. However, this performance comes

with a cost of slower recognition speed. To address this issue, we introduced Tesla-V, a faster prediction model, with only 1.5% drop in accuracy while performing inference 3 times faster than Tesla. Thus, Tesla and Tesla-V cover a wide range of applications with different speed-accuracy requirements.

Egocentric Applications. Due to the computational efficiency and robustness to different environments and angles, Tesla-Rapture system can be extended to scenarios where egocentric gestures should be recognized on constrained devices. Tesla prediction model can be modified to adapt to new applications for wearable devices, e.g., Microsoft HoloLens[3]. Currently, HoloLens 2 captures hand gestures using RGB-D sensors. Given the benefits of radars over RGB-D cameras (see section 2), integration of Tesla-Rapture with HoloLens improves the performance of hand gesture recognition which is one of the main interaction mechanisms of this device.

Tesla on Dense Point Clouds. We trained and evaluated Tesla on SHREC-28 [68] and NVGesture [11] datasets, in both of which a set of dense gestures are collected using a depth camera. The proposed model achieves 81.5% and 80.1% accuracy, while the state of the art (PointLSTM) has 94.7% and 87.9% accuracy on SHREC-28 and NVGesture, respectively suggesting that Tesla fails to capture spatial structures in each frame effectively which is vital for dense point cloud processing. Since Tesla aims at recognizing gestures from mmWave radar generated point clouds, highly sparse compared to that of other devices (see section 3.3), capturing spatial features of each frame does not contribute to the performance. Our approach outperforms PointLSTM (state of the art model on SHREC-28 and NVGesture), with a margin of up to 12.4%, 11.1%, and 4.6% accuracy on mmWave radar generated point cloud datasets of Pantomime, mHomeGes, and RadHAR, respectively (see section 6.4.1).

Tesla on PRM (Pantomime, RadHAR, mHomeGes). To further analyze the performance of the proposed model, Tesla, on scenarios with higher number of gestures, we combined the three datasets to have 36 classes of gestures. Tesla was trained from scratch on 70% of the data and tested using the rest. The accuracy is 93.31% indicating that the model is able to generalize well on the scenarios with more number of classes.

Future Work. While in this work, we introduced Temporal Graph K-NN to reflect the temporal dependency in graph generation, the graph is still being created statically using Graph K-NN algorithm. Reinforcement Learning (RL), imitating the cognitive reward based learning process, enhances the graph according to the accuracy of the classification model. Therefore, dynamic graph generation using RL is one possible direction for improving the temporal graph.

8 Conclusion

In this work, we proposed Tesla-Rapture, a real-time gesture recognition interface based on mmWave radar generated sparse point clouds. In doing so, we designed Temporal Graph K-NN to implicitly reflect the temporal evolution of gestures in a temporal graph on which the proposed attention-based MPNN is applied to recognize gestures. Moreover, we presented two versions of Tesla and Tesla-V employing the mentioned strategy. Our results show that Tesla-Rapture enhances the accuracy up to 21% in extreme settings while reducing the prediction time by a magnitude of 8 and computational complexity (GFLOPs) by almost 40 times compared to the most accurate competitor.

3. https://www.microsoft.com/en-us/hololens

Acknowledgment

We thank the anonymous referees for the constructive feedback provided. Part of the calculations presented above were performed using computer resources within the Aalto University School of Science "Science-IT" project.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant agreement No. 813999.

References

[1] Q. Wan, Y. Li, C. Li, and R. Pal, "Gesture recognition for smart home applications using portable radar sensors," in *2014 36th annual international conference of the IEEE engineering in medicine and biology society*. IEEE, 2014, pp. 6414–6417.

[2] E. Ohn-Bar and M. M. Trivedi, "Hand gesture recognition in real time for automotive interfaces: A multimodal vision-based approach and evaluations," *IEEE transactions on intelligent transportation systems*, vol. 15, no. 6, pp. 2368–2377, 2014.

[3] H. Liu and L. Wang, "Gesture recognition for human-robot collaboration: A review," *International Journal of Industrial Ergonomics*, vol. 68, pp. 355–367, 2018.

[4] K. Kalgaonkar and B. Raj, "One-handed gesture recognition using ultrasonic doppler sonar," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2009, pp. 1889–1892.

[5] R. J. Przybyla, H.-Y. Tang, S. E. Shelton, D. A. Horsley, and B. E. Boser, "12.1 3d ultrasonic gesture recognition," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. IEEE, 2014, pp. 210–211.

[6] Z. Lu, X. Chen, Q. Li, X. Zhang, and P. Zhou, "A hand gesture recognition framework and wearable gesture-based interaction prototype for mobile devices," *IEEE transactions on human-machine systems*, vol. 44, no. 2, pp. 293–299, 2014.

[7] Y. Zhang and C. Harrison, "Tomo: Wearable, low-cost electrical impedance tomography for hand gesture recognition," in *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, 2015, pp. 167–173.

[8] P. Gupta, K. Kautz *et al.*, "Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural networks," in *CVPR*, vol. 1, no. 2, 2016, p. 3.

[9] P. K. Pisharady and M. Saerbeck, "Recent methods and databases in vision-based hand gesture recognition: A review," *Computer Vision and Image Understanding*, vol. 141, pp. 152–165, 2015.

[10] X. Yang, P. Molchanov, and J. Kautz, "Making convolutional networks recurrent for visual sequence learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6469–6478.

[11] P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree, and J. Kautz, "Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4207–4215.

[12] M. Abavisani, H. R. V. Joze, and V. M. Patel, "Improving the performance of unimodal dynamic hand-gesture recognition with multimodal training," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[13] Y. Kim and B. Toomajian, "Hand gesture recognition using micro-doppler signatures with convolutional neural network," *IEEE Access*, vol. 4, pp. 7125–7130, 2016.

[14] C. R. Qi, H. Su, K. Mo, and 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.

[15] Y. Min, Y. Zhang, X. Chai, and X. Chen, "An efficient pointlstm for point clouds based gesture recognition," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 5760–5769.

[16] S. Palipana, D. Salami, L. A. Leiva, and S. Sigg, "Pantomime: Mid-air gesture recognition with sparse millimeter-wave radar point clouds," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 5, no. 1, pp. 1–27, 2021.

[17] D. Salami, S. Palipana, M. Kodali, and S. Sigg, "Motion pattern recognition in 4d point clouds," in *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2020, pp. 1–6.

[18] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 945–953.

[19] T. Yu, J. Meng, and J. Yuan, "Multi-view harmonized bilinear network for 3d object recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 186–194.

[20] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 922–928.

[21] G. Riegler, A. Osman Ulusoy, and A. Geiger, "Octnet: Learning deep 3d representations at high resolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3577–3586.

[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," *Acm Transactions On Graphics (tog)*, vol. 38, no. 5, pp. 1–12, 2019.

[23] H. U. Odoemelem and K. Van Laerhoven, "A low-cost prototyping framework for human-robot desk interaction," in *Adjunct Proceedings of the 2020 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2020 ACM International Symposium on Wearable Computers*, 2020, pp. 191–194.

[24] V. D. Soni, "Artificial cognition for human-robot interaction," *International Journal on Integrated Education*, vol. 1, no. 1, pp. 49–53, 2018.

[25] V. Vujović and M. Maksimović, "Raspberry pi as a sensor web node for home automation," *Computers & Electrical Engineering*, vol. 44, pp. 153–171, 2015.

[26] S. Jain, A. Vaibhav, and L. Goyal, "Raspberry pi based interactive home automation system through e-mail," in *2014 International Conference on Reliability Optimization and Information Technology (ICROIT)*. IEEE, 2014, pp. 277–280.

[27] J. P. Wachs, M. Kölsch, H. Stern, and Y. Edan, "Vision-based hand-gesture applications," *Commun. ACM*, vol. 54, no. 2, 2011.

[28] S. S. Rautaray and A. Agrawal, "Vision based hand gesture recognition for human computer interaction: A survey." *Artif. Intell. Rev.*, vol. 43, no. 1, pp. 1–54, 2012.

[29] R. Lun and W. Zhao, "A survey of applications and human motion recognition with Microsoft Kinect," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 29, no. 5, pp. 1 555 008:1–48, 2015.

[30] K. Caine, S. Šabanovic, and M. Carter, "The effect of monitoring by cameras and robots on the privacy enhancing behaviors of older adults," in *Proc. HRI*, 2012, pp. 343–350.

[31] H. Abdelnasser, M. Youssef, and K. A. Harras, "WiGest: A ubiquitous wifi-based gesture recognition system," in *Proc. INFOCOM*, 2015.

[32] H. Li, W. Yang, J. Wang, Y. Xu, and L. Huang, "WiFinger: talk to your smart devices with finger-grained gesture," in *Proc. UbiComp*, 2016, pp. 250–261.

[33] Y. Ma, G. Zhou, S. Wang, H. Zhao, and W. Jung, "SignFi: Sign language recognition using wifi," *ACM IMWUT*, vol. 2, no. 1, pp. 1–21, 2018.

[34] R. H. Venkatnarayan, G. Page, and M. Shahzad, "Multi-user gesture recognition using WiFi," in *Proc. MobiSys*, 2018, pp. 401–413.

[35] A. Virmani and M. Shahzad, "Position and orientation agnostic gesture recognition using WiFi," in *Proc. MobiSys*, 2017, pp. 252–264.

[36] Q. Pu, S. Gupta, S. Gollakota, and S. Patel, "Whole-home gesture recognition using wireless signals," in *Proc. MobiCom*, 2013, pp. 27–38.

[37] T. Li, L. Fan, M. Zhao, Y. Liu, and D. Katabi, "Making the invisible visible: Action recognition through walls and occlusions," in *Proc. ICCV*, 2019, pp. 872–881.

[38] M. Zhao, Y. Liu, A. Raghu, T. Li, H. Zhao, A. Torralba, and D. Katabi, "Through-wall human mesh recovery using radio signals," in *Proc. ICCV*, 2019, pp. 10 113–10 122.

[39] J. Lien, N. Gillian, M. E. Karagozler, P. Amihood, C. Schwesig, E. Olson, H. Raja, and I. Poupyrev, "Soli: Ubiquitous gesture sensing with millimeter wave radar," *ACM Trans. Graphics*, vol. 35, no. 4, pp. 1–19, 2016.

[40] T. Wei and X. Zhang, "mTrack: High-precision passive tracking using millimeter wave radios," in *Proc. MobiCom*, 2015, pp. 117–129.

[41] A. D. Berenguer, M. C. Oveneke, H. Khalid, M. Alioscha-Perez, A. Bourdoux, and H. Sahli, "GestureVLAD: Combining unsupervised features representation and spatio-temporal aggregation for doppler-radar gesture recognition," *IEEE Access*, vol. 7, pp. 137 122–137 135, 2019.

[42] P. S. Santhalingam, A. A. Hosain, D. Zhang, P. Pathak, H. Rangwala, and R. Kushalnagar, "Environment-Independent ASL Gesture Recognition Using 60 GHz Millimeter-wave Signals," *ACM IMWUT*, vol. 4, no. 1, pp. 1–30, 2020.

[43] A. D. Singh, S. S. Sandha, L. Garcia, and 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*, 2019, pp. 51–56.

[44] S. Wang, J. Song, J. Lien, I. Poupyrev, and O. Hilliges, "Interacting with Soli: Exploring fine-grained dynamic gesture recognition in the radio-frequency spectrum," in *Proc. UIST*, 2016, pp. 851–860.

[45] 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 *In Proc. of DCOSS*. IEEE, 2019, pp. 33–40.

[46] Z. Meng, S. Fu, J. Yan, H. Liang, A. Zhou, S. Zhu, H. Ma, J. Liu, and N. Yang, "Gait recognition for co-existing multiple people using millimeter wave sensing," in *In Proc. of AAAI*, vol. 34, no. 01, 2020, pp. 849–856.

[47] C. Jiang, J. Guo, Y. He, M. Jin, S. Li, and Y. Liu, "mmvib: micrometer-level vibration measurement with mmwave radar," in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, 2020, pp. 1–13.

[48] K. Qian, Z. He, and X. Zhang, "3d point cloud generation with millimeter-wave radar," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, no. 4, pp. 1–23, 2020.

[49] Z. Dong, F. Li, J. Ying, and K. Pahlavan, "A model-based rf hand motion detection system for shadowing scenarios," *IEEE Access*, vol. 8, pp. 115 662–115 672, 2020.

[50] H. Liu, Y. Wang, A. Zhou, H. He, W. Wang, K. Wang, P. Pan, Y. Lu, L. Liu, and H. Ma, "Real-time arm gesture recognition in smart home scenarios via millimeter wave sensing," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, no. 4, pp. 1–28, 2020.

[51] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17, 2017.

[52] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *Proceedings of the 5th International Conference on Learning Representations*, ser. ICLR '17, 2017.

[53] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17, 2017.

[54] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *ACM Trans. Graph.*, vol. 38, no. 5, Oct. 2019. [Online]. Available: https://doi.org/10.1145/3326362

[55] J. Owoyemi and K. Hashimoto, "Spatiotemporal learning of dynamic gestures from 3d point cloud data," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–5.

[56] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[57] M. A. Richards, J. A. Scheer, and W. A. Holm, *Principles of Modern Radar: Basic Principles*. Scitech Publishing, 2010.
[58] M. Jaderberg, K. Simonyan, A. Zisserman, and k. kavukcuoglu, "Spatial transformer networks," in *Advances in Neural Information Processing Systems*, vol. 28. Curran Associates, Inc., 2015, pp. 2017–2025.
[59] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017, pp. 5998–6008.
[60] G. Cohen, M. Hilario, H. Sax, S. Hugonnet, and A. Geissbuhler, "Learning from imbalanced data in surveillance of nosocomial infection," *Artificial intelligence in medicine*, vol. 37, no. 1, pp. 7–18, 2006.
[61] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *arXiv preprint arXiv:1912.01703*, 2019.
[62] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," *arXiv preprint arXiv:1903.02428*, 2019.
[63] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[64] J. Ruiz and Y. Li, "Doubleflip: a motion gesture delimiter for mobile interaction," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2011, pp. 2717–2720.
[65] F. Kerber, P. Schardt, and M. Löchtefeld, "Wristrotate: a personalized motion gesture delimiter for wrist-worn devices," in *Proceedings of the 14th international conference on mobile and ubiquitous multimedia*, 2015, pp. 218–222.
[66] F. Masina, V. Orso, P. Pluchino, G. Dainese, S. Volpato, C. Nelini, D. Mapelli, A. Spagnolli, and L. Gamberini, "Investigating the accessibility of voice assistants with impaired users: Mixed methods study," *Journal of medical Internet research*, vol. 22, no. 9, p. e18431, 2020.
[67] H. Fan, Y. Yang, and M. Kankanhalli, "Point 4d transformer networks for spatio-temporal modeling in point cloud videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 14 204–14 213.
[68] Q. De Smedt, H. Wannous, J.-P. Vandeborre, J. Guerry, B. L. Saux, and D. Filliat, "3d hand gesture recognition using a depth and skeletal dataset: Shrec'17 track," in *Proceedings of the Workshop on 3D Object Retrieval*, 2017, pp. 33–38.

Dariush Salami received his BSc and MSc degrees from Shahid Beheshti University and Amirkabir University of Technology in Software Engineering in 2016 and 2019, respectively. He is currently a Marie Skłodowska Curie fellow in ITN-WindMill project and a PhD researcher at the department of communications and networking at Aalto University. He is mainly focused on Machine Learning for Wireless Communications and Sensing especially in mmWave range.

Ramin Hasibi received his the BSc and MSc from Isfahan University of Technology and Amirkabir University of Technology in Information Technology Engineering in 2016 and 2019, respectively. He is currently a Ph.D. researcher at the department of informatics, University of Bergen where his main research focus is on Graph Representation Learning and Graph Neural Networks as well as their application in different domains.

Sameera Palipana is currently a System Specification Engineer at Nokia Solutions and Networks, Espoo and a Visiting Researcher at Aalto University. He was a Postdoctoral Researcher at Aalto University between 2019-2021. He obtained his PhD from Munster Technological University, Ireland, in 2019, received his Master's degree at University of Bremen, Germany in 2014 in Information and Communication Technology, and received his B.Sc. (Hons) degree in Electronics and Telecommunication Engineering from University of Moratuwa, Sri Lanka in 2010.

Petar Popovski is a Professor at Aalborg University, where he heads the section on Connectivity and a Visiting Excellence Chair at the University of Bremen. He received his Dipl.-Ing and M. Sc. degrees in communication engineering from the University of Sts. Cyril and Methodius in Skopje and the Ph.D. degree from Aalborg University in 2005. He is a Fellow of the IEEE. He received an ERC Consolidator Grant (2015), the Danish Elite Researcher award (2016), IEEE Fred W. Ellersick prize (2016), IEEE Stephen O. Rice prize (2018), Technical Achievement Award from the IEEE Technical Committee on Smart Grid Communications (2019), the Danish Telecommunication Prize (2020) and Villum Investigator Grant (2021). He is a Member at Large at the Board of Governors in IEEE Communication Society, Vice-Chair of the IEEE Communication Theory Technical Committee and IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING. He is currently an Area Editor of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS. Prof. Popovski was the General Chair for IEEE SmartGridComm 2018 and IEEE Communication Theory Workshop 2019. His research interests are in the area of wireless communication and communication theory. He authored the book "Wireless Connectivity: An Intuitive and Fundamental Guide", published by Wiley in 2020.

Tom Michoel is Professor in bioinformatics at the Computational Biology Unit at the Department of Informatics at the University of Bergen since 2018, and was an independent group leader in computational biology at the University of Edinburgh (2012-2018) and the University of Freiburg (2010-2012). He obtained the MSc degree in Physics (1997) and PhD degree in Mathematical Physics (2001) from the KU Leuven, and was a postdoctoral researcher in mathematics (UC Davis, 2001-2002), theoretical physics (KU Leuven, 2002-2004), and bioinformatics and systems biology (Ghent University, 2004-2010). His research focus in the last five years has been on developing methods, algorithms, and software for causal inference and Bayesian network learning from high-dimensional omics data, supported by grants from the BBSRC (2015-2016), the NIH (2016-2019), the MRC (2017-2021), and the Norwegian Research Council (2021-2024).

Stephan Sigg received his M.Sc. degree in computer science from TU Dortmund, in 2004 and his Ph.D. degree from Kassel University, in 2008. Since 2015 he is an assistant professor at Aalto University, Finland. He is a member of the editorial board of the Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies as well as of the Elsevier journal of Computer Communications. He has served as a TPC member of renowned conferences including IEEE PerCom, IEEE ICDCS, etc. His research interests include Ambient Intelligence, in particular, Pervasive sensing, activity recognition, usable security algorithms for mobile distributed systems.