General spiking neural network framework for the learning trajectory from a noisy mmWave radar

NEUROMORPHIC
Computing and Engineering

OPEN ACCESS

RECEIVED
26 February 2022

REVISED
19 June 2022

ACCEPTED FOR PUBLICATION
10 August 2022

PUBLISHED
15 September 2022

PAPER

General spiking neural network framework for the learning trajectory from a noisy mmWave radar

Xin Liu[1,2], Mingyu Yan[1,2,*], Lei Deng[3,*], Yujie Wu[4], De Han[1,2], Guoqi Li[2,5], Xiaochun Ye[1,2] and Dongrui Fan[1,2]

[1] SKLP, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, People's Republic of China
[2] University of Chinese Academy of Sciences, Beijing, People's Republic of China
[3] Department of Precision Instrument, Center for Brain Inspired Computing Research (CBICR), Tsinghua University, Beijing, People's Republic of China
[4] Institute of Theoretical Computer Science, Graz University of Technology, Graz, Austria
[5] Institute of Automation, Chinese Academy of Sciences, Beijing, People's Republic of China
[*] Author to whom any correspondence should be addressed.

E-mail: yanmingyu@ict.ac.cn and leideng@mail.tsinghua.edu.cn

Keywords: millimeter wave radar, spiking neural networks, attention mechanism, model robustness, trajectory estimation

Abstract

Emerging usages for millimeter wave (mmWave) radar have drawn extensive attention and inspired the exploration of learning mmWave radar data. To be effective, instead of using conventional approaches, recent works have employed modern neural network models to process mmWave radar data. However, due to some inevitable obstacles, e.g., noise and sparsity issues in data, the existing approaches are generally customized for specific scenarios. In this paper, we propose a general neuromorphic framework, termed mm-SNN, to process mmWave radar data with spiking neural networks (SNNs), leveraging the intrinsic advantages of SNNs in processing noisy and sparse data. Specifically, we first present the overall design of mm-SNN, which is adaptive and easily expanded for multi-sensor systems. Second, we introduce general and straightforward attention-based improvements into the mm-SNN to enhance the data representation, helping promote performance. Moreover, we conduct explorative experiments to certify the robustness and effectiveness of the mm-SNN. To the best of our knowledge, mm-SNN is the first SNN-based framework that processes mmWave radar data without using extra modules to alleviate the noise and sparsity issues, and at the same time, achieve considerable performance in the task of trajectory estimation.

1. Introduction

In recent years, diverse applications in measurement and estimation have been rapidly emerging in industrial scenarios. As a promising detection and ranging system, millimeter wave (mmWave) radars have attracted substantial attention in solving various tasks [3, 8, 12, 26, 27, 31, 39]. Generally, a mmWave radar operates in the frequency spectrum from 30 to 300 GHz. By transmitting an electromagnetic signal and receiving the return signal, it measures the surrounding information of the target object that contains relative speed, range, and angle, even if in a harsh environment [12]. However, processing and learning from mmWave radars is non-trivial. The point cloud generated by a mmWave radar inevitably contains outliers. Such outliers are irregularly distributed in the point cloud space and are located far away from the target object, which contributes nothing but noise to the following data processing [39]. Moreover, the generated point cloud is usually sparse, making the processing more difficult. In addition, considering a common combination of the mmWave radar and other systems, such as a visual sensing system, in a complex task [3], the temporal dependency between multiple systems needs to be handled well during modeling.

Typically, it is common to process mmWave radar data using signal processing approaches [2, 36]. Nevertheless, due to the intrinsic properties of mmWave radars and the increasing data volume, conventional approaches gradually present limitations. As a growing trend, deep learning based approaches are proposed to process mmWave radar data in recent literature [8, 12, 26, 31]. For example, milliMap [26] leverages a generative adversarial network [11] to address the noise and sparsity issues, in which a conditional generator is designed to denoise and densify image patches, and a discriminator is designed to distinguish the real and generated images. mm-Pose [31] uses convolutional neural networks (CNNs) [24] to map the encoded RGB images obtained from the mmWave radar into a high-dimensional space. Unfortunately, existing deep learning based methods for processing mmWave radar data are often customized for specific scenarios [8]. In addition, the temporal dependency in mmWave radar data is challenging to model. mmMesh [39] introduces a long short-term memory (LSTM) [17] model and adopts global/local LSTM layers to encode the temporal information, which brings huge parameters and a high computation cost. Moreover, extra costs and modules are required for dealing with the noise and sparsity issues in mmWave radar data. Therefore, a generic approach to solve all the above challenges is highly expected.

Spiking neural networks (SNNs), known as the third generation of neural networks [28], exploit the rich spatial-temporal dynamics of neuromorphic neurons to represent the complex information in spike patterns. Specifically, each spiking neuron in an SNN model continuously integrates weighted spikes from afferent neurons along the temporal dimension onto the membrane potential, and fires a spike event to efferent neurons once its membrane potential crosses a threshold. In this way, SNNs have natural advantages in capturing the temporal dependency of input data [6]. Moreover, SNNs are superior in handling noisy and sparse features, owing to the robust leakage and firing mechanisms of spiking neurons [4, 6, 16]. The above unique features of SNNs imply that it is promising to build a general framework with SNNs for processing the temporal, noisy, and sparse mmWave radar data, which is the goal and focus of this work.

To this end, we first propose mm-SNN, a general neuromorphic framework for processing mmWave radar data with SNN models, which resists the defects of noise and sparsity while effectively capturing the temporal dependency. Then, we introduce multiple attention-based mechanisms into the proposed framework to further improve the data representation and model accuracy. At last, we conduct extensive experiments to evaluate the proposed framework by processing real-world mmWave radar data in a trajectory estimation scenario. Specifically, the contributions of this paper can be summarized as follows:

- We propose mm-SNN, a general neuromorphic framework for processing mmWave radar data with SNN models. mm-SNN is the first work that processes mmWave radar data without using extra modules for alleviating noise and sparsity issues.
- To demonstrate the effectiveness and robustness of mm-SNN, we artificially add noises in the inputs of both mm-SNN and non-spiking models. The experimental results reveal the robustness of mm-SNN.
- We apply mm-SNN to the scenario of trajectory estimation and introduce multiple attention-based mechanisms to improve the overall performance of mm-SNN. We find that the general mechanisms used in deep learning, such as attention, are also able to benefit the performance of SNNs by enhancing the data representation.

The rest of this paper is organized as follows. In section 2, we first introduce the background of the mmWave radar. Then, we indicate the present obstacles in learning mmWave radar data caused by the noise and sparsity issues. To solve the challenges, we further highlight the great potential of SNNs that can be fully leveraged to process mmWave radar data. In section 3, we first introduce the foundation of SNN models. Then, we propose the design of our general neuromorphic framework, i.e., mm-SNN, for processing mmWave radar data. Based on the proposed framework, we introduce attention-based mechanisms to improve the data representation and model accuracy. In section 4, we conduct extensive experiments along with comprehensive comparisons and analyses to demonstrate the effectiveness and robustness of mm-SNN. Finally, we conclude this paper in section 5.

2. Preliminary

2.1. Backgrounds of mmWave radar

mmWave is a particular electromagnetic wave whose wavelength lies in 1 to 10 mm. The property of short-wavelength increases the potential of frequency reuse and makes the mmWave a wide use in military projects. A mmWave radar transmits signals that are in the short-wavelength range and operates in the high-frequency bands. It generally utilizes techniques, such as frequency modulated continuous wave [33], to emit and receive signals for measuring the surrounding information of the target object in a 3D environment. Using short-wavelength signals, mmWave radars lower the size requirement of system components (e.g., antennas) and enhance the ability to detect highly tiny movements [22]. Recently, mmWave radars have been actively applied to many scenarios, owing to their inherent advantage of propagating in a harsh environment. Prior work have explored employing the usage of mmWave radars in diverse applications such as human sensing [31, 39] and environment sensing [8, 26]. Moreover, in some industrial scenarios, a mmWave radar can be tentatively utilized as a critical component for determining the distance and angle in the automotive industry [9, 21], since it works well at night and in environments with airborne floatage (e.g., smoke, fog, and dust) [10, 12, 26]. Achievements of adopting mmWave radars in both academia and industry thus prove their popularity.

2.2. Challenges in processing mmWave radar data

mmWave radars show promising performance in tasks of measurement and estimation. However, due to the intrinsic properties, processing and learning from mmWave radar data is still a challenge. Herein, we list two typical properties, i.e., **noise** and **sparsity** in mmWave data. Moreover, problems caused by these properties and some possible solutions are also highlighted as follows.

Noise. As a long-standing issue, noise in data can be generated for a variety of factors. For instance, multi-path noise usually occurs in the process of light spread and reflection [26, 27]. Unfortunately, the noise, especially some outliers in data, can lead to non-negligible problems. As the noisy data is fed into a neural network based framework for learning, outliers in the input data can deteriorate the model training by causing gradient explosion. Existing works have spent considerable efforts to deal with the noise or remove outliers in data. mmMesh [39] proposes to remove the noisy mmWave signals reflected by static surrounding objects via a pre-processing component. SuperRF [8] proposes to design a compressed sensing based approach to reduce the influence of the noise on model output accuracy. However, the denoising process usually brings about an extra cost that cannot always be neglected, especially when the model size is not large enough to cover the preprocessing cost.

Sparsity. Distinct from other techniques, such as Light Detection and Ranging (LiDAR), a mmWave radar generally generates highly sparse data. For instance, a typical LiDAR scan has 100$\times$ more points than a mmWave radar in the same situation, which means the point cloud generated by a mmWave radar is 100$\times$ sparser than a LiDAR [26]. Such a sparsity property can cause several problems. First, processing a large size point cloud where only a few points are valid will bring about a unnecessary computational cost [31]. Second, sparsity can degrade the quality of data, thus invalidating conventional methods that are designed for processing LiDAR data [26]. Previous work have explored solving the sparsity issue via various approaches such as point cloud upsampling [42] and attention mechanisms [39]. Despite these existing solutions, dealing with the sparsity issue of mmWave radar data still needs a case-by-case analysis.

2.3. Potential of SNNs

The aforementioned properties, i.e., the noise and sparsity in data, cause challenges in the downstream processing and learning. Moreover, many existing approaches proposed to solve the noise and sparsity issues are generally tailored for specific scenarios. For example, SuperRF [8] customizes its method for the radio frequency (RF) sensing scenario considering the distinction between RF signals and the ordinary point clouds. Thereby, given a case where multiple types of sensory data (more than only mmWave radar data) can be flexibly introduced for modelling, it is expected to have all data processed by a general framework, instead of designing case-by-case modules for each type of data.

Brain-inspired SNNs have drawn increasing attention, owing to their biological plausibility and model robustness. As the third generation of neural network models, SNNs work in a neuro-biomimetic approach, in which each neuron decides to update the membrane potential or fire a spike according to the received input, the memorized state, and the difference between the membrane potential and the threshold. By leveraging the strengths of SNNs, some existing works have achieved significant success in several applications [5, 30, 40, 43]. Recently, there has been emerging research interest in exploring the intrinsic properties of SNNs, especially the robustness of SNNs. LISNN [4] suppresses the noisy area in the input and improves the anti-noise ability of the SNN model by adding a lateral interaction mechanism. HIRE-SNN [23] leverages a timing dependent backpropagation approach to train an SNN model and enhance its intrinsic robustness. It is also discussed in prior literature [6, 16] that SNNs can use a self-neuron recurrence restriction mechanism to restrict the weight update for resisting the impact from noise and stabilizing the SNN model.

Since the point cloud generated by a mmWave radar is sparse and generally contains noise, learning from such data will inevitably face two challenges: extracting and learning features from the sparse input, and dealing with noise in the input data. As aforementioned, SNNs have natural advantages in processing noisy data, which is benefited from the leakage and firing mechanisms. Moreover, SNNs use sparse spike activities to represent information and show promising potential in processing sparse features [16]. The above characteristics provide opportunities for learning mmWave radar data via an SNN-based framework.

3. Methodology

3.1. Fundamentals of SNN

The leaky integrate and fire (LIF) model is one of the commonly used neural models for describing the behaviors of a spiking neuron, such as input integration, membrane potential update, and spike generation. In artificial neural networks (ANNs), the ReLU function (and the variant leaky ReLU) is used for activation. Similarly, SNNs use the LIF model to determine whether to update the membrane potential or reset the potential and fire a spike simultaneously. Applying the LIF model to SNNs has the following advantages: (1) it is tractable in the aspect of mathematical calculation. (2) It preserves the biological characteristics of neurons in the brain to a certain extent (3) it improves the robustness of SNN models against external noise with the help of leakage and firing mechanisms. Typically, we describe a differential format of the LIF model as follows:

$$\begin{aligned} \tau\frac{\mathrm{d}u(t)}{\mathrm{d}t} &= -u(t) + I(t), \quad u(t) < u_{\text{threshold}}, \\ \text{Fire a spike \& Reset } u(t)&, \quad u(t) \geqslant u_{\text{threshold}}, \end{aligned} \tag{1}$$

where $u(t)$ denotes the postsynaptic membrane potential that varies with t, $I(t)$ denotes the presynaptic input. τ is a time constant and $u_{\text{threshold}}$ is the preset threshold for firing spikes. Apparently, according to equation (1), the membrane potential $u(t)$ is updated based on the input $I(t)$ and the historic state of $u(t)$. In another case, the target neuron fires a spike, and the membrane potential $u(t)$ is reset to a preset value. Moreover, to make the LIF model more explicit and easier for implementation on existing deep learning frameworks (e.g., Tensorflow [1], Pytorch [29]), we adopt an iterative version proposed in the literature [38]. The equations we used to implement the LIF model in SNNs are governed by

$$u^{n+1,i}(t+1) = k_{\text{decay}}u^{n+1,i}(t)(1 - o^{n+1,i}(t)) + \sum_{j=1} w^{n,ij}o^{n,j}(t+1), \tag{2}$$

$$o^{n+1,i}(t+1) = H(u^{n+1,i}(t+1) - u_{\text{threshold}}), \tag{3}$$

where more specific representations are added to the original LIF model, that is, $u^{n,i}(t)$ denotes the membrane potential of the ith neuron in the nth layer, $o^{n,i}(t)$ denotes the spiking activity (i.e., 1 for firing and 0 for not) of the ith neuron in the nth layer. Additionally, k_{decay} is a decay factor and $w^{n,ij}$ is the synaptic weight. The function $H(\cdot)$ is a Heaviside function that determines whether firing a spike event or not by evaluating the relationship of amplitude between the membrane potential $u^{n+1,i}(t+1)$ and the threshold $u_{\text{threshold}}$. In the case of $u^{n+1,i}(t+1) \geqslant u_{\text{threshold}}$, the output of $H(\cdot)$ equals 1, i.e., firing a spike event.

With respect to the training scheme, we adopt the spatio-temporal backpropagation (STBP) [37] for SNNs. In STBP, the gradients, i.e., $\frac{\mathrm{d}L}{\mathrm{d}u_t^{n,i}}$ and $\frac{\mathrm{d}L}{\mathrm{d}o_t^{n,i}}$ (L denotes the loss function) are needed to compute for the model update. However, $\frac{\mathrm{d}o_t^{n,i}}{\mathrm{d}u_t^{n,i}}$ does not exist due to the nondifferentiable nature of the spiking activity. A surrogate function is introduced to approximate the derivative of $H(\cdot)$:

$$\frac{\mathrm{d}o}{\mathrm{d}u} \approx \begin{cases} \frac{1}{a}, & |u - u_{\text{threshold}}| \leqslant \frac{a}{2} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where u denotes the membrane potential, and a is an adjustment coefficient for controlling the membrane potential window. Equation (4) indicates that the gradient is allowed to pass through only the membrane potential is close the threshold.

3.2. General SNN framework

(1) *Overall framework:* figure 1 illustrates the overall framework of mm-SNN. Taking raw data as the input, mm-SNN outputs the learned representation that can be directly used in downstream tasks. Systematically, we divide the overall process into the following five substeps:

❶ Processing data: the raw data is preprocessed via approaches such as projection and discretization [25]. After preprocessing, the data can be directly processed by SNN models.

❷ Learning data: in mm-SNN, we choose suitable SNN models for diverse types of processed data. For instance, a 3D point cloud is converted into the 2D structured format that includes abundant information in the spatial dimension, just like images. Inspired by the successful attempt of estimating optical flow through an SNN model [14], we explicitly construct a temporal relationship between two continuous batches of the 2D data (i.e., $t = T$ and $t = T + 1$) to capture the temporal dependency.

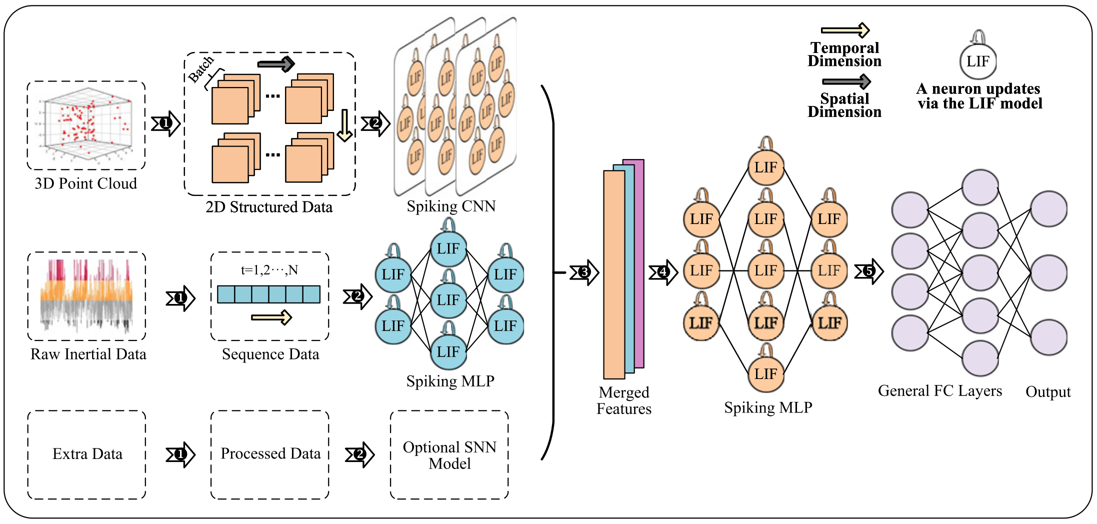

Figure 1. Illustration of the overall framework. The input of mm-SNN is the raw data captured from multiple sensors, and the output of mm-SNN can be directly used for specific applications, implying that mm-SNN works in an end-to-end manner.

❸ Merging features: given the features of diverse data that have been extracted by SNN models, to fit the design of a multi-sensor system, mm-SNN synthetically utilizes multiple features to generate a global feature for modeling temporal dependency. The detailed process will be introduced in section 3.
❹ Modeling temporal dependency: temporal dependency lies in two situations. In one sensory data, such as mmWave radar data, the relationship in the temporal dimension of two continuous batches has been captured by a spiking CNN model. Furthermore, since multiple features are merged into a global one, the temporal dependency between these features needs to be modeled via a spiking multi-layer perceptron (MLP).
❺ Transformation: the use of fully connected (FC) layers in mm-SNN follows the general idea of using FC in CNNs for feature transformation and classification. The features generated by the spiking MLP are transformed into a low-dimensional space for output.

The information from a mmWave radar can be supplemented by the information from other sensors to form high-quality data for downstream tasks [13, 31]. As an example, two sensory data, i.e., a point cloud generated by a mmWave radar and an inertial sequence generated by an inertial sensor, are fed to the SNN model for learning. Please note that, as a general framework, the architecture of mm-SNN is flexible to extend to handle extra sensors by simply introducing SNN modules for processing the data from those sensors.
(2) *Dependency in temporal dimension:* temporal dependency is critical in learning information among multiple relative objects in the temporal dimension. We explain the detailed approaches and the significance of modeling the temporal dependency in two situations as follows. First, as aforementioned, to resist the impact of the sparsity issue in the point cloud, we utilize a projection function to convert the 3D point coordinates into the corresponding 2D positions to form the 2D structured data. Since point cloud data is widely used in object detection and motion estimation tasks, we thereby start with some attempts in the computer vision domain to help extract useful representation from the 2D structured data. Inspired by some CNN-based approaches [7, 20] that take two images as the respective inputs of two identical branch networks, we take the concept of optical flow [18] and construct the relationship in the temporal dimension of two continuous 2D data (i.e., $t = T$ and $t = T + 1$). Instead of processing two images via a branched CNN model, we combine two continuous 2D data as an integrated input and feed it into a spiking CNN. The spiking CNN is qualified to extract spatial-temporal features from the structured 2D data, owing to the excellent ability of capturing temporal dependency. Second, to promote generality, we design the mm-SNN to fit a multi-sensor system, in which more than one sensors are utilized to collect data from one target object. For example, a mmWave radar detects the motion attitude and orientation, while an inertial sensor measures the rotation and acceleration. The recorded point cloud and sequences are simultaneously used to estimate the overall motion situation. In this case, it is critical to capture the temporal dependency among data generated by multiple sensors, especially for the merged features. Therefore, we adopt a spiking MLP to model the temporal dependency of the features after merging. A spiking MLP generally has a shallow network architecture, fewer parameters, and enough ability of capturing temporal dependency, helping achieve comparable performance in temporal modeling with a lower cost.

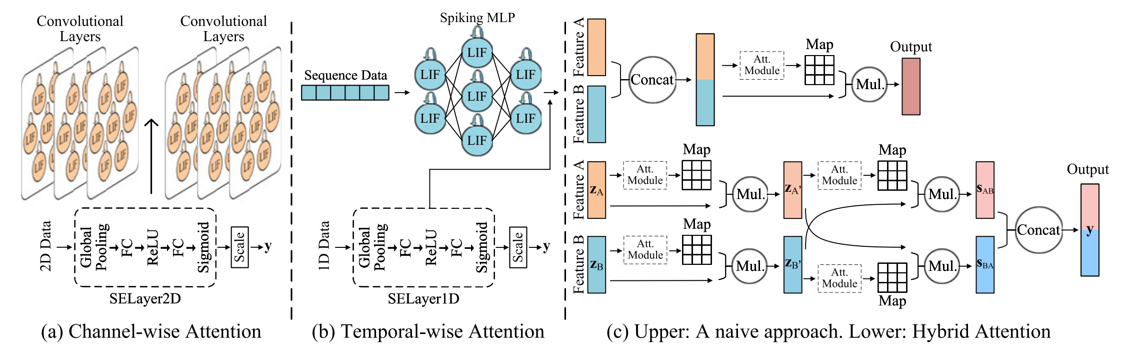

Figure 2. Improvements via attention mechanisms: (a) SELayer2D, an attention module that focuses on the channel-wise information and can be directly plugged into a spiking CNN; (b) SELayer1D, an attention module that focuses on the temporal-wise information and can be directly plugged after a spiking MLP; (c) (upper) a naive approach for handling multiple features, and (lower) a preferable approach that leverages hybrid attention for fusing multiple features.

3.3. Improvements: leveraging attention mechanisms

Targeting processing mmWave radar data and other sensory data in a general manner is the fundamental design purpose of mm-SNN. By adopting diverse SNN models, mm-SNN is able to effectively learn from inputs generated by a multi-sensor system and yields outputs for downstream tasks. Moreover, we argue that the performance of mm-SNN can be further improved by adding extra mechanisms. Herein, we choose the attention mechanism as a case study owing to its wide usage, considerable performance, and 'plug and play' convenience. To be universal, we introduce general and straightforward attention-based improvements into mm-SNN to enhance the data representation during learning. Please note that, we merely introduce some typical attention-based mechanisms, while it is of great potential to leverage more advanced attention modules for further improvement. We demonstrate the performance of adding attention-based mechanisms in section 2.

(1) *Channel-wise attention module for 2D structured data:* SENet [19] was first proposed to adaptively recalibrate weights among channels and capture the channel-wise relationship. The core mechanism of SENet is the ingenuity of a squeeze-and-excitation (SE) block. Specifically, the squeeze module first applies a global average pooling operation to gather spatial information. Then, the excitation module captures the channel-wise dependency by utilizing a stacked structure of FC layers and non-linear activation functions (ReLU and Sigmoid). The above process can be formulated as

$$z_c = \mathtt{Squeeze}(\mathbf{x}_c) = \frac{1}{H \times W} \sum_{j=1}^{H} \sum_{k=1}^{W} x_c(j,k), \tag{5}$$

$$\mathbf{s} = \mathtt{Excitation}(\mathbf{W}, \mathbf{z}) = \sigma(\mathbf{W}_2 \delta(\mathbf{W}_1 \mathbf{z})), \tag{6}$$

$$\mathbf{y}_c = \mathtt{Scale}(s_c, \mathbf{x}_c) = s_c \cdot \mathbf{x}_c, \tag{7}$$

where $\mathbf{x}_c$ denotes the input feature map in the shape of $\mathbb{R}^{H\times W}$, $H \times W$ is the spatial resolution. $\mathbf{W}_1$ and $\mathbf{W}_2$ denote learnable parameters for reducing computational complexity. $\sigma(\cdot)$ and $\delta(\cdot)$ denote non-linear activation functions, i.e., ReLU and Sigmoid. The final output $\mathbf{y}_c$ is acquired by adopting the channel-wise multiplication between $\mathbf{x}_c$ and the scalar s_c. As illustrated in figure 2(a), the module can be plugged between two continuous layers in a spiking CNN. The reason for choosing this attention module is two-fold: (1) SENet is a classic mechanism that pioneers the channel attention for improving the representation ability; (2) since the preprocessing operation projects the unstructured point cloud to the 2D structured data for further learning, we adopt a spiking CNN to extract features from such data, which imitates the case of processing images via a typical non-spiking CNN. Therefore, we argue that the typical attention mechanism, i.e., SENet, can also enhance the representation in spiking models, and refer to such module as SELayer2D in the rest of this paper.

(2) *Temporal-wise attention module for sequences:* a temporal-wise attention module for SNNs is recently proposed in TA-SNN [41] to improve the classification performance of event streams. Execution of the module can also be divided into two steps, i.e., the squeeze step and the excitation step, taking inspiration from the SENet [19]. The squeeze step calculates a statistical vector $\mathbf{s}^{n-1}$ of event numbers for each timestep t in the nth layer by using a global average pooling operation. The excitation step feds the statistical vector into a two-layer FC model with nonlinear activation functions to obtain the correlation vector $\mathbf{d}^{n-1}$ between different frames.

The above process can be formulated as

$$s_t^{n-1} = \frac{1}{C \times H \times W} \sum_{c=1}^{C} \sum_{i=1}^{H} \sum_{j=1}^{W} \mathbf{X}_t^{n-1}(c, i, j), \tag{8}$$

$$\mathbf{d}^{n-1} = \begin{cases} \sigma(\mathbf{W}_2^n \delta(\mathbf{W}_1^n \mathbf{s}^{n-1})), & \text{training}, \\ f(\sigma(\mathbf{W}_2^n \delta(\mathbf{W}_1^n \mathbf{s}^{n-1})) - d_{\text{th}}), & \text{inference}, \end{cases} \tag{9}$$

where $\mathbf{X}_t^{n-1}$ denotes the spatial input tensor in the nth layer at the tth timestep, and C denotes the channel size. $\mathbf{W}_1^n$ and $\mathbf{W}_2^n$ denote learnable parameters in the nth layer. $f(\cdot)$ denotes a unit step function, $\sigma(\cdot)$ and $\delta(\cdot)$ are the same as those in equation (6). d_{th} is a threshold parameter used to adjust $\mathbf{d}^{n-1}$ in the inference phase. In the situation of processing 1D sequences, only the training phase is focused. As illustrated in figure 2(b), we modify the above process by performing an adaptive global average pooling on the sequence data, in which the temporal dimension is preserved for modeling interdependency. Specifically, we implement the module using a similar structure of SELayer2D, and we refer to the temporal-wise attention module as SELayer1D in the rest of this paper.

(3) *Hybrid attention module for merged features*: hybrid attention is inspired by recent literature [27, 34], in which multiple attention modules are combined to acquire focused areas in merged features. Nevertheless, hybrid attention is not the only approach to handle features captured from multiple sensors. As illustrated in figure 2(c), a naive approach is that features can be directly concatenated before being fed into an attention module. Since features extracted from multiple sensors are concatenated as a whole, using a single attention module has limitations when multi-dimension (i.e., spatial and temporal dimensions) dependencies abundantly exist between various types of features. Therefore, we use the hybrid attention, a combination of two single attention (function named `Att`$(\cdot)$) and two cross attention (function named `CrossAtt`$(\cdot)$) modules, to establish a relationship between various types of features. We formulate the procedures of performing the hybrid attention as

$$\mathbf{z}_{A'} = \texttt{Att}(\mathbf{z}_A) = \sigma(\mathbf{W}_2 \delta(\mathbf{W}_1 \mathbf{z}_A)) \otimes \mathbf{z}_A, \tag{10}$$

$$\mathbf{z}_{B'} = \texttt{Att}(\mathbf{z}_B) = \sigma(\mathbf{W}_4 \delta(\mathbf{W}_3 \mathbf{z}_B)) \otimes \mathbf{z}_B, \tag{11}$$

$$\mathbf{s}_{AB} = \texttt{CrossAtt}(\mathbf{z}_{A'}, \mathbf{z}_{B'}) = \sigma(\mathbf{W}_6 \delta(\mathbf{W}_5 \mathbf{z}_{A'})) \otimes \mathbf{z}_{B'}, \tag{12}$$

$$\mathbf{s}_{BA} = \texttt{CrossAtt}(\mathbf{z}_{B'}, \mathbf{z}_{A'}) = \sigma(\mathbf{W}_8 \delta(\mathbf{W}_7 \mathbf{z}_{B'})) \otimes \mathbf{z}_{A'}, \tag{13}$$

$$\mathbf{y} = \texttt{Concat}(\mathbf{s}_{AB}, \mathbf{s}_{BA}) = \mathbf{s}_{AB} \oplus \mathbf{s}_{BA}, \tag{14}$$

where $\mathbf{z}_A$ and $\mathbf{z}_B$ denote the feature A and B extracted by a spiking CNN and a spiking MLP, respectively. $\otimes$ denotes the element-wise multiplication between two matrices, and $\oplus$ denotes the feature-wise concatenation between two matrices. Notably, definitions of $\mathbf{W}_n$, $\sigma(\cdot)$, and $\delta(\cdot)$ are the same as those in equation (6). The output $\mathbf{y}$ is used to model the temporal dependencies between various types of features in a spiking MLP.

4. Experiments

4.1. Data preparation

To verify the effectiveness and robustness of mm-SNN, we conduct experiments to estimate the trajectory using a real-world dataset [27]. The dataset consists of 49 sequences for training and 8 sequences for testing. The raw data is collected by a single-chip mmWave radar (TI AWR1843) and a commercial-grade inertial measurement unit (IMU). As for one sequence, it is filled with the preprocessed data from the mmWave radar and IMU. Specifically, the mmWave radar data in the dataset has been preprocessed as follows:

$$\theta = \text{atan2}(y, x), \tag{15}$$

$$\phi = \arcsin(z/\sqrt{x^2 + y^2 + z^2}), \tag{16}$$

$$r = \lfloor \theta/\Delta\theta \rfloor, \tag{17}$$

$$c = \lfloor \phi/\Delta\phi \rfloor, \tag{18}$$

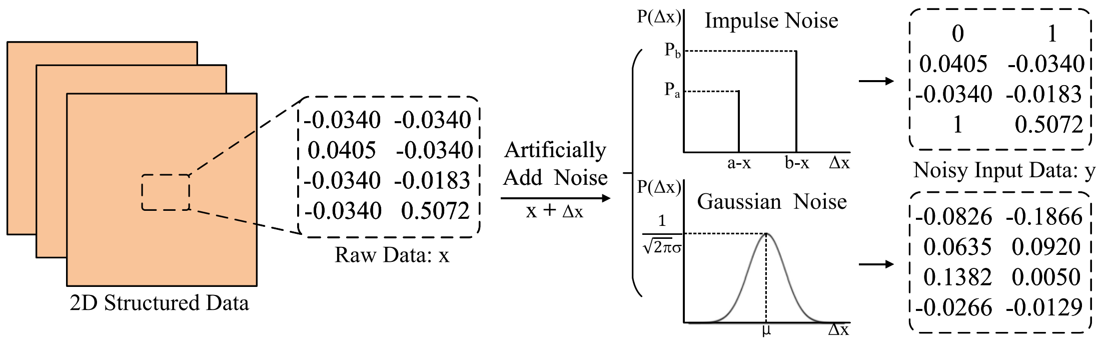

Figure 3. Illustration of artificially modified inputs with impulse noise and Gaussian noise added. We add noises into the 2D structured data respectively.

where the 2D map position (r, c) is projected from (x, y, z) points in the 3D space. θ and ϕ are the azimuth and elevation angle through point observation. $\Delta\theta$ and $\Delta\phi$ are the average horizontal and vertical angle resolution between consecutive beam emitters. More details please refer to [25]. Following [27], a normalization technique is applied to the 2D data after projection.

Despite the preprocessing, we discover that the properties of mmWave radar data are generally unchanged by analyzing the processed data. We load the mmWave radar data that have been converted to the 2D data and analyze such data in an element-wise manner. By adopting normalization, values of (pixel-level) elements in the 2D data are standardized into the range of $(-1, 1)$. However, it is observed that a value $v \approx -0.0340$ repeatedly exists in each sequence. We count the number of the highly repeated values, i.e., v, and quantify the proportion of v among all elements of the 2D data in all sequences. Surprisingly, v takes an 84.19% proportion of all training sequences. Similar to the sparsity issue in the 3D cloud point, we argue that a mass of repeated values in the 2D data will generally weaken the ability of feature representation and pattern learning, which brings about challenges in processing such data.

Moreover, to demonstrate the performance of handling noisy data, two types of noises are artificially added into the input for testing. Specifically, as illustrated in figure 3, we add the impulse noise and the Gaussian noise into the projected 2D structured data. The raw data is regarded as the variable x, and the noise is defined as Δx. The noisy input data y can be obtained by adding x and Δx. Descriptions of the above noises are formulated as follows:

$$P(\Delta x) = \frac{1}{\sqrt{2\pi}\sigma}\,\mathrm{e}^{-(\Delta x-\mu)^2/2\sigma^2}, \tag{19}$$

$$P(\Delta x) = \begin{cases} P_a, & \Delta x = a - x \ (\text{i.e.}, x + \Delta x = a) \\ P_b, & \Delta x = b - x \ (\text{i.e.}, x + \Delta x = b) \\ 1 - P_a - P_b, & \Delta x = 0 \ (\text{i.e., without noise}) \end{cases}. \tag{20}$$

Herein, since values in the 2D space have been normalized into the range of $(-1, 1)$, we modify the setting of parameters to a reasonable range when generating the noises. For the impulse noise, we set the parameters a and b to 0 and 1 respectively. The probability parameters P_a and P_b are flexible to adjust. In this case, we define an ever-changing random value p between 0 and 1. For each (pixel-level) element in the 2D data, we set the corresponding element to zero when p is less than P_a, and set the corresponding element to one when p is larger than $(1 - P_b)$. For other cases $(P_a \leqslant p \leqslant 1 - P_b)$, the corresponding element remains unchanged. For the Gaussian noise, we set the parameter σ to 0.01. μ is adjustable and is set to be an increasing parameter from 0 to reflect the growing intensity of Gaussian noise. We directly add the generated Gaussian noise in the 2D structured data. For convenience, we refer to the input with impulse noise and Gaussian noise as the impulse-noisy input and the Gaussian-noisy input to distinguish the original input in the rest of the paper.

4.2. Experimental setting

(1) *Benchmark and setting:* to validate the effectiveness of mm-SNN, we conduct comparisons between the predicted trajectories and the ground truth. We introduce the absolute trajectory error (ATE) to quantify the accuracy of an entire trajectory [15]. The quality of the trajectory estimation can be quantified by statistic metrics of ATE, e.g., root mean square error (RMSE) [32], mean, median, standard deviation (Std.), and Max error. Specifically, the experiments are organized as follows: in section 1, we focus on the entire process of the model learning of mm-SNN and a baseline inspired by [27], i.e., a non-spiking framework, and plot the curves of the

Table 1. Model structures used in mm-SNN.

Model	Network structure
Spiking CNN	C64_RF7-C128_RF5-C256_RF5-C256_RF3-C512_RF3-C512_RF3-C512_RF3-C512_RF3-C1024_RF3
Spiking MLP_1	C128_Linear
Spiking MLP_2	C1024_Linear-C512_Linear

test loss evolving with the training epoch; in section 2, we conduct an ablation study to explore how attention-based modules benefit the vanilla mm-SNN; in section 3, we utilize ATE to quantify the accuracy of prediction between mm-SNN and the non-spiking framework; in section 4, we conduct experiments with different noise levels to compare the capability of noise resistance between mm-SNN and the non-spiking framework. Notice that the non-spiking framework consists of non-spiking models which share the same training configurations (see section 3) and basic network structures with those in mm-SNN.
(2) *Structures of models:* to perform the task of trajectory estimation, we specify some simple yet efficient structures for SNN models. Specifically, we utilize the basic configurations (e.g., number of channels, receptive field sizes) of a CNN model proposed in [35]. Our spiking CNN model mainly distinguishes from a typical CNN model at the mechanism of neuronal computation. Instead of direct propagating after a nonlinear activation function, each neuron in the spiking CNN model performs an update based on the LIF behaviors. Moreover, the operation of batch normalization is used at the end of each layer to benefit the training. The spiking CNN model is leveraged in mm-SNN to extract features and learn representation from the mmWave radar data. In addition to the spiking CNN model, we utilize a one-layer spiking MLP model (named Spiking MLP_1) to process the sequence data of IMU, and a two-layer spiking MLP model (named Spiking MLP_2) to model the temporal dependency of the merged features. Note that SNNs' structures are specified for our targeted task and are flexible to change according to the scenario where mm-SNN is adopted. Detailed information on model structures can be found in table 1. To be clear, 'C64_RF7' denotes a convolutional layer whose number of channels is 64 and receptive field size is 7×7, and 'C128_Linear' denotes a linear layer whose number of channels is 128.
(3) *Training configurations:* the training of the overall models is performed under the following configurations. The batch size and the maximum number of epochs are set to 64 and 100, respectively. In the condition that the performance promotion is slight, the general training permits an early stop to ensure a preferable generalization ability and a lower time cost. We use the root mean square prop optimizer with a dynamically adjusted learning rate to tune the training. The learning rate is initialized to 1×10^{-5} with a regular decay by 75% per 25 epochs. Moreover, we highlight the setting of hyper-parameters used in SNN models. The threshold $u_{\text{threshold}}$ used in the LIF model in equation (1) is set to 0.4. The decay factor used in the membrane update in equation (2) is set to 0.2. The coefficient a used for approximating the derivative of spike activity in equation (4) is set to 1. We implement all models in mm-SNN in Pytorch [29]. All experiments are conducted on a Linux server equipped with dual 14-core Intel Xeon E5-2683 v3 CPUs and an NVIDIA Tesla V100 GPU (16 GB memory).

4.3. Result and analysis

(1) *Robustness to noise:* regular data generally contains abundant spatial information and is easy to process. The mmWave radar data, however, is more complicated even after projection. As aforementioned, two obstacles exist in the learning of preprocessed mmWave radar data: (1) a specific value repeatedly exists in all sequences; (2) noise is artificially added into the input to estimate the ability of resisting noise. Without the intrinsic property of robustness, we argue that the above obstacles could deteriorate the model learning of conventional ANNs. We thereby conduct experiments between a non-spiking framework and mm-SNN. For both frameworks, all models are trained with the original input and are tested with three types of inputs, i.e., original input, the impulse-noisy input, and the Gaussian-noisy input. For fairness, models in the non-spiking framework and mm-SNN share the same network structures. For example, mm-SNN uses a spiking CNN to extract features from the mmWave radar data, while the non-spiking framework leverages a typical non-spiking CNN with the same structure as the spiking CNN in mm-SNN for identical use. Moreover, the capability of modeling temporal dependency is unique to spiking neurons, but not to non-spiking ones. Therefore, we introduce an LSTM in the non-spiking framework. With respect to the learning schemes, the non-spiking framework and mm-SNN are trained by the ANN-oriented backpropagation through time algorithm and the SNN-oriented STBP algorithm, respectively.

We visualize the model learning process by recording the test loss evolving by epoch. As illustrated in figure 4, given three types of inputs, the curves of the test loss of mm-SNN consistently show a stable and quick downtrend. However, for the non-spiking framework, the curves of the test loss are obviously unstable and fluctuant, revealing an unsatisfied performance on model generalization. Notably, it is observed that many leap points exist in the curves of the non-spiking framework, especially for models tested with noisy

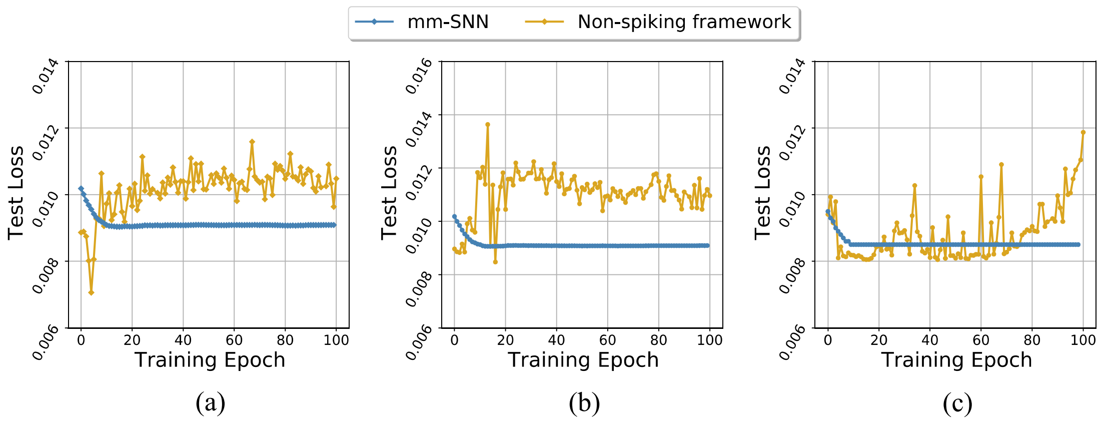

Figure 4. Comparison of test curves: (a) the variation of the test loss between the non-spiking framework and mm-SNN, with original input as the test data; (b) the variation of the test loss between the non-spiking framework and mm-SNN, with the impulse-noisy input as the test data; (c) a similar comparison with the impulse noise substituted by the Gaussian noise. For the impulse-noisy input, the parameters P_a and P_b are all set to 0.1. For the Gaussian-noisy input, the parameters μ and σ are set to 0.5 and 0.01, respectively.

inputs (see figures 4(b) and (c)). Herein, the leap point means a loss point with a nontrivial leap compared to the overall tendency. We argue that the unstable curves of the non-spiking framework are blamed for the disturbance in data, e.g., the two obstacles in the mmWave radar data. In addition, we discover that different noisy input affects models in the non-spiking framework to different degrees. The impulse-noisy input generally has a greater effect on models in the non-spiking framework than the Gaussian-noisy input by comparing the overall trend of curves of the test loss. On the other hand, with the intrinsic advantages in resisting noises, SNN models in mm-SNN are more robust to handle the noisy inputs, making the model learning stable. Specifically, the robustness of mm-SNN might be benefited from spiking neurons with leakage, firing, and reset mechanisms, which can naturally act as a noise filter. In addition, it is observed that injecting the Gaussian noise into the sequence can help the feature learning (see figures 4(a) and (c)). As aforementioned, a value $v \approx -0.0340$ repeatedly appears in each sequence. Injecting the Gaussian noise would change the imbalanced data distribution in a sequence to a more stochastic distribution, thus yielding a slight improvement in accuracy.

(2) *Ablation study on attention-based improvements:* we visualize the predicted trajectories and the corresponding trajectories of ground truth to highlight the gap using four distinct models, i.e., 'mm-SNN', the vanilla mm-SNN framework equipped with a channel-wise attention module ('W. SELayer2D'), the vanilla mm-SNN framework equipped with a temporal-wise module ('W. SELayer1D'), the vanilla mm-SNN framework equipped with a hybrid attention module ('W. HybridAtt.') in figure 5. In addition to attention modules, all models are trained under the same configuration. We choose the best models within 100 training epochs for visualization. From an overall perspective, 'mm-SNN' achieves superior prediction performance since the predicted trajectories precisely fit the ground truth in most cases. From the perspective of the ablation study, the prediction performance of the vanilla mm-SNN equipped with a single attention module varies by test sequences. It is observed that 'W. SELayer2D' achieves a sub-optimal performance compared to 'mm-SNN' on three test sequences (see figures 5(a), (c) and (d)). In addition, 'W. SELayer1D' performs well in the other three test sequences (see figures 5(b), (c) and (e)). Unfortunately, the prediction performance of 'W. HybridAtt.' is not comparable on any test sequence. However, only 'mm-SNN' shows good prediction performance on all test sequences, which is difficult to achieve for all vanilla mm-SNN frameworks equipped with one attention module merely.

Next, we discuss the reason for distinguishing prediction performance among models with different attention modules equipped. As mentioned in section 3, the channel-wise attention module (SELayer2D) and the temporal-wise attention module (SELayer1D) are plugged into SNN models, i.e., the spiking CNN and the spiking MLP, for extracting features, while the hybrid attention module is utilized in the process of feature fusion. We argue that an attention module plugged into a feature extractor is better for enhancing the data representation of features. In consequence, the analysis on the ablation study reveals three facts for us to design a framework with multiple SNN models integrated: (1) Vanilla SNN models can be improved via attention-based modules. (2) A framework equipped with more applicable attention-based modules can gain better performance overall. (3) The optimization effect of an attention-based module is distinct by the plugged location.

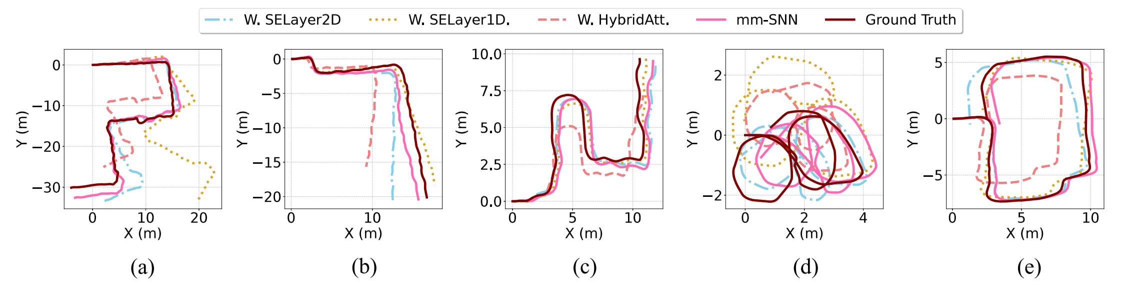

Figure 5. Comparisons among predicted trajectories and the ground truth. Subplots (a)–(e) denote five distinct test sequences. In all subplots, 'mm-SNN' denotes the proposed framework with all attention modules equipped. 'W. SELayer2D' denotes the vanilla mm-SNN framework with merely the channel-wise attention module, i.e., SELayer2D, equipped. Similarly, 'W. SELayer1D' and 'W. HybridAtt.' denote the vanilla mm-SNN framework with temporal-wise attention and hybrid attention modules equipped, respectively.

Table 2. Quantified analysis on statistic metrics of ATE. The lower these values, the better performance in general. The row named 'average' denotes the averaged result among five test sequences using the same models (in the non-spiking framework or mm-SNN) for quantification. The lowest error quantified by the five metrics on each test sequence is set in **bold** for convenience.

Test data	Model	Sequence	RMSE	Mean	Median	Std.	Max
Impulse-noisy input	Non-spiking framework	Test Seq. 1	25.7434	20.1859	16.3460	15.9766	55.5138
		Test Seq. 2	6.8999	5.0490	3.2079	4.7028	16.3836
		Test Seq. 3	3.1194	2.4471	1.4886	1.9345	5.9314
		Test Seq. 4	0.6401	0.5744	0.4891	0.2825	1.3621
		Test Seq. 5	3.2286	2.8724	2.9408	1.4743	5.3434
		Average	7.9263	6.2257	4.8945	4.8741	16.9069
	mm-SNN	Test Seq. 1	2.3166	1.6939	1.3617	1.5803	5.9049
		Test Seq. 2	2.7535	2.3125	2.3321	1.4946	4.8256
		Test Seq. 3	0.6855	0.6515	0.6001	0.2131	1.1990
		Test Seq. 4	0.6954	0.6499	0.6270	0.2474	1.6544
		Test Seq. 5	1.0635	0.9779	1.1652	0.4180	1.5884
		Average ★	1.5029	1.2571	1.2172	0.7907	3.0344
Gaussian-noisy input	Non-spiking framework	Test Seq. 1	26.0004	20.3956	16.7588	16.1258	55.9151
		Test Seq. 2	6.4851	4.6652	2.7420	4.5048	15.6540
		Test Seq. 3	2.6071	1.9871	1.3903	1.6877	4.6410
		Test Seq. 4	0.7916	0.6863	0.5219	0.3945	1.7236
		Test Seq. 5	3.2586	2.9492	3.2516	1.3859	5.1572
		Average	7.8286	6.1367	4.9329	4.8197	16.6182
	mm-SNN	Test Seq. 1	3.9502	3.2775	3.1745	2.2050	8.0452
		Test Seq. 2	0.5780	0.4657	0.4580	0.3424	1.5530
		Test Seq. 3	0.6459	0.6215	0.5480	0.1760	0.9336
		Test Seq. 4	0.9181	0.8597	0.8352	0.3221	1.6290
		Test Seq. 5	0.7653	0.7168	0.7436	0.2680	1.1705
		Average ★	1.3715	1.1882	1.1519	0.6627	2.6663

(3) *Quantified comparison and analysis:* we adopt ATE to quantify the accuracy of prediction between mm-SNN and the non-spiking framework with five statistic metrics: RMSE, mean, median, Std., and Max. The performance on different test sequences and the averaged results are given in table 2. To reveal the effect of noisy inputs on the model performance, we test models in the non-spiking framework and mm-SNN with two types of inputs, i.e., the impulse-noisy input and the Gaussian-noisy input. For the impulse-noisy input, the parameters P_a and P_b are set to 0.01 and 0.01. For the Gaussian-noisy input, the parameters μ and σ are set to 0.01 and 0.01. We choose the best models within 100 training epochs for quantifying the prediction performance. It is first observed that mm-SNN achieves the lowest error on all metrics from an overall perspective (see rows named 'Average' marked with ★), revealing the superior performance of mm-SNN against noisy inputs on most test sequences. For instance, using the impulse-noisy input, the RMSE metric of mm-SNN is about 2 (on 'Test Seq. 2') to 11 (on 'Test Seq. 1') times lower than that of the non-spiking framework. Although the gap is slight, we also find situations that sometimes mm-SNN cannot significantly outperform the non-spiking framework (e.g., prediction on 'Test Seq. 4'). The reason that mm-SNN cannot always perform superiorly on all test sequences might be that each test sequence is independently collected and generated with distinct characteristics. In consequence, quantified results undoubtedly demonstrate a superior performance of mm-SNN from an overall perspective (see rows named 'Average' marked with ★) and better performance on most test sequences.

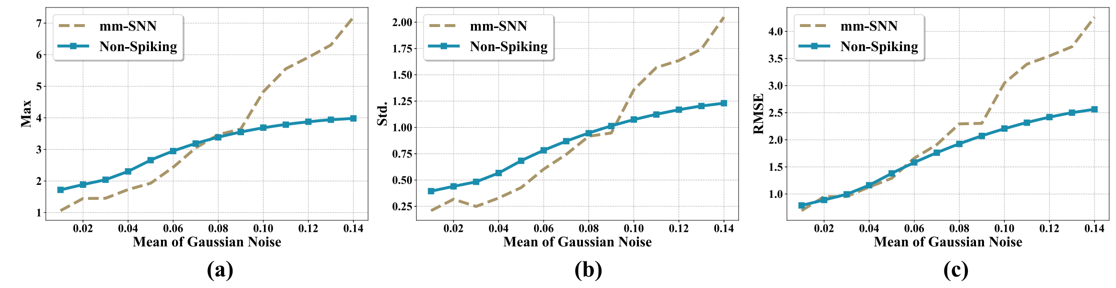

Figure 6. Performance comparison between mm-SNN and the non-spiking framework under different noise levels.

(4) *Study on noise resistance:* we further conduct experiments to compare the capability of noise resistance between mm-SNN and the non-spiking framework. Specifically, we first add a Gaussian noise to one test sequence and gradually increase the noise level by changing mean (the parameter μ) of the Gaussian noise. Then, we quantify the statistic metrics on the same test sequence when running mm-SNN and the non-spiking framework. We reminder that an increase in any of the metrics indicates a decrease in the prediction performance of a model.

As depicted in figure 6, for both mm-SNN and the non-spiking framework, the model performance degrades as the noise level grows. Different metrics generally show different increments under the same noise increment. We discover that the Max and Std. metrics in the non-spiking framework are larger than those in mm-SNN, given the noise level is lower than 0.09 (figures 6(a) and (b)). When the noise level further increases, the Max and Std. metrics in mm-SNN surpass those in the non-spiking framework, indicating a significant degradation in the prediction performance. The RMSE metric presents a similar trend although the metrics are closer before the noise level approaches 0.06. The above observations inspire the following interesting conclusion. Compared to the smooth degradation of the non-spiking framework, mm-SNN is more robust until the noise increases to a certain level. Since the state space of SNNs is discrete and each LIF neuron is a natural noise filter, the states are harder to be changed by injecting a small noise but easier to collapse once the noise level becomes high enough.

5. Conclusion

This paper presents a general neuromorphic framework, named mm-SNN, to process the mmWave radar data via SNN-based models. The proposed framework is robust to noisy inputs, and is scalable for multi-sensor systems. Moreover, we find that the generic mechanisms for data enhancement used in deep learning can also provide considerable improvement in performance of mm-SNN. Extensive experiments are conducted to demonstrate the effectiveness and robustness of mm-SNN. mm-SNN is not only a paradigm to learn representation from the mmWave radar data but also provides an end-to-end solution to flexibly handle data with temporal dependency acquired by a complex system.

Acknowledgments

This work was partly supported by National Natural Science Foundation of China (Grant Nos. 62106119, 61732018 and 61872335), Austrian-Chinese Cooperative R&D Project (FFG and CAS) (Grant No. 171111KYSB20200002), CAS Project for Young Scientists in Basic Research (Grant No. YSBR-029), and CAS Project for Youth Innovation Promotion Association.

Data availability statement

All data that support the findings of this study are included within the article (and any supplementary files).

ORCID iDs

Xin Liu https://orcid.org/0000-0003-2147-3870

References

[1] Abadi M *et al* 2016 Tensorflow: a system for large-scale machine learning *12th USENIX Symp. Operating Systems Design and Implementation (OSDI 16)* pp 265–83
[2] Bhatia J *et al* 2021 Object classification technique for mmWave FMCW radars using range-FFT features *Int. Conf. COMmunication Systems & NETworkS (COMSNETS)* (Piscataway, NJ: IEEE) pp 111–5
[3] Chang S, Zhang Y, Zhang F, Zhao X, Huang S, Feng Z and Wei Z 2020 Spatial attention fusion for obstacle detection using mmWave radar and vision sensor *Sensors* **20** 956
[4] Cheng X, Hao Y, Xu J and Xu B 2020 LISNN: improving spiking neural networks with lateral interactions for robust object recognition *IJCAI* pp 1519–25
[5] Deng L *et al* 2020 Tianjic: a unified and scalable chip bridging spike-based and continuous neural computation *IEEE J. Solid-State Circuits* **55** 2228–46
[6] Deng L, Wu Y, Hu X, Liang L, Ding Y, Li G, Zhao G, Li P and Xie Y 2020 Rethinking the performance comparison between SNNS and ANNS *Neural Netw.* **121** 294–307
[7] Dosovitskiy A, Fischer P, Ilg E, Hausser P, Hazirbas C, Golkov V, Van Der Smagt P, Cremers D and Brox T 2015 Flownet: learning optical flow with convolutional networks *Proc. IEEE Int. Conf. Computer Vision* pp 2758–66
[8] Fang S and Nirjon S 2020 SuperRF: enhanced 3D RF representation using stationary low-cost mmWave radar *Int. Conf. Embedded Wireless Systems and Networks (EWSN) 2020* (NIH Public Access) pp 120–31
[9] Ford 2020 Adaptive cruise control https://ford.com/technology/driver-assist-technology/adaptive-cruise-control/
[10] Garcia K, Yan M and Purkovic A 2018 *Robust Traffic and Intersection Monitoring Using Millimeter Wave Sensors* (Dallas, TX: Texas Instruments)
[11] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A and Bengio Y 2014 Generative adversarial nets *Advances in Neural Information Processing Systems* vol 27
[12] Guan J, Madani S, Jog S, Gupta S and Hassanieh H 2020 Through fog high-resolution imaging using millimeter wave radar *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition* pp 464–73
[13] Guo X-p, Du J-s, Gao J and Wang W 2018 Pedestrian detection based on fusion of millimeter wave radar and vision *Proc. 2018 Int. Conf. Artificial Intelligence and Pattern Recognition* pp 38–42
[14] Haessig G, Cassidy A, Alvarez R, Benosman R and Orchard G 2018 Spiking optical flow for event-based sensors using IBM's TrueNorth neurosynaptic system *IEEE Trans. Biomed. Circuits Syst.* **12** 860–70
[15] Handa A, Whelan T, McDonald J and Davison A J 2014 A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM *2014 IEEE Int. Conf. Robotics and Automation (ICRA)* (Piscataway, NJ: IEEE) pp 1524–31
[16] He W, Wu Y, Deng L, Li G, Wang H, Tian Y, Ding W, Wang W and Xie Y 2020 Comparing SNNS and RNNS on neuromorphic vision datasets: similarities and differences *Neural Netw.* **132** 108–20
[17] Hochreiter S and Schmidhuber J 1997 Long short-term memory *Neural Comput.* **9** 1735–80
[18] Horn B K and Schunck B G 1981 Determining optical flow *Artif. Intell.* **17** 185–203
[19] Hu J, Shen L and Sun G 2018 Squeeze-and-excitation networks *Proc. IEEE Conf. Computer Vision and Pattern Recognition* pp 7132–41
[20] Ilg E, Mayer N, Saikia T, Keuper M, Dosovitskiy A and Brox T 2017 Flownet 2.0: evolution of optical flow estimation with deep networks *Proc. IEEE Conf. Computer Vision and Pattern Recognition* pp 2462–70
[21] T Instruments 2021 Automotive mmWave radar sensors https://ti.com/sensors/mmwave-radar/automotive/overview.html
[22] Iovescu C and Rao S 2017 *The Fundamentals of Millimeter Wave Sensors* (Dallas, TX: Texas Instruments) pp 1–8
[23] Kundu S, Pedram M and Beerel P A 2021 Hire-SNN: harnessing the inherent robustness of energy-efficient deep spiking neural networks by training with crafted input noise *Proc. IEEE/CVF Int. Conf. Computer Vision* pp 5209–18
[24] LeCun Y *et al* 1995 Convolutional networks for images, speech, and time series *The Handbook of Brain Theory and Neural Networks* vol 3361 (Cambridge, MA: MIT Press)
[25] Li B, Zhang T and Xia T 2016 Vehicle detection from 3D Lidar using fully convolutional network (arXiv:1608.07916)
[26] Lu C X, Rosa S, Zhao P, Wang B, Chen C, Stankovic J A, Trigoni N and Markham A 2020 See through smoke: robust indoor mapping with low-cost mmWave radar *Proc. 18th Int. Conf. Mobile Systems, Applications, and Services* pp 14–27
[27] Lu C X, Saputra M R U, Zhao P, Almalioglu Y, de Gusmao P, Chen C, Sun K, Trigoni N and Markham A 2020 milliego: single-chip mmWave radar aided egomotion estimation via deep sensor fusion *Proc. 18th Conf. Embedded Networked Sensor Systems* pp 109–22
[28] Maass W 1997 Networks of spiking neurons: the third generation of neural network models *Neural Netw.* **10** 1659–71
[29] Paszke A *et al* 2019 Pytorch: an imperative style, high-performance deep learning library *Advances in Neural Information Processing Systems* vol 32 pp 8026–37
[30] Pei J *et al* 2019 Towards artificial general intelligence with hybrid Tianjic chip architecture *Nature* **572** 106–11
[31] Sengupta A, Jin F, Zhang R and Cao S 2020 mm-Pose: real-time human skeletal posture estimation using mmWave radars and CNNS *IEEE Sens. J.* **20** 10032–44
[32] Sturm J, Magnenat S, Engelhard N, Pomerleau F, Colas F, Cremers D, Siegwart R and Burgard W 2011 Towards a benchmark for RGB-D SLAM evaluation *RGB-D Workshop on Advanced Reasoning with Depth Cameras at Robotics: Science and Systems Conf. (RSS)*
[33] Uttam D and Culshaw B 1985 Precision time domain reflectometry in optical fiber systems using a frequency modulated continuous wave ranging technique *J. Lightwave Technol.* **3** 971–7
[34] Wang L, Wang Y, Liang Z, Lin Z, Yang J, An W and Guo Y 2019 Learning parallax attention for stereo image super-resolution *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition* pp 250–9
[35] Wang S, Clark R, Wen H and Trigoni N 2017 Deepvo: towards end-to-end visual odometry with deep recurrent convolutional neural networks *2017 IEEE Int. Conf. Robotics and Automation (ICRA)* (Piscataway, NJ: IEEE) pp 2043–50
[36] Wu X, Zhang N, Zhang H and Hong W 2016 Frequency estimation algorithm for ranging of millimeter wave LFMCW radar *2016 IEEE Int. Conf. Ubiquitous Wireless Broadband (ICUWB)* (Piscataway, NJ: IEEE) pp 1–3

[37] Wu Y, Deng L, Li G, Zhu J and Shi L 2018 Spatio-temporal backpropagation for training high-performance spiking neural networks *Front. Neurosci.* **12** 331

[38] Wu Y, Deng L, Li G, Zhu J, Xie Y and Shi L 2019 Direct training for spiking neural networks: faster, larger, better *Proc. AAAI Conf. Artificial Intelligence* vol 33 pp 1311–8

[39] Xue H, Ju Y, Miao C, Wang Y, Wang S, Zhang A and Su L 2021 mmMesh: towards 3D real-time dynamic human mesh construction using millimeter-wave *Proc. 19th Annual Int. Conf. Mobile Systems, Applications, and Services* pp 269–82

[40] Yan Y *et al* 2021 Comparing Loihi with a SpiNNaker 2 prototype on low-latency keyword spotting and adaptive robotic control *Neuromorphic Comput. Eng.* **1** 014002

[41] Yao M, Gao H, Zhao G, Wang D, Lin Y, Yang Z and Li G 2021 Temporal-wise attention spiking neural networks for event streams classification *Proc. IEEE/CVF Int. Conf. Computer Vision* pp 221–30

[42] Yu L, Li X, Fu C-W, Cohen-Or D and Heng P-A 2018 Pu-net: point cloud upsampling network *Proc. IEEE Conf. Computer Vision and Pattern Recognition* pp 2790–9

[43] Zhang X *et al* 2020 An artificial spiking afferent nerve based on Mott memristors for neurorobotics *Nat. Commun.* **11** 51