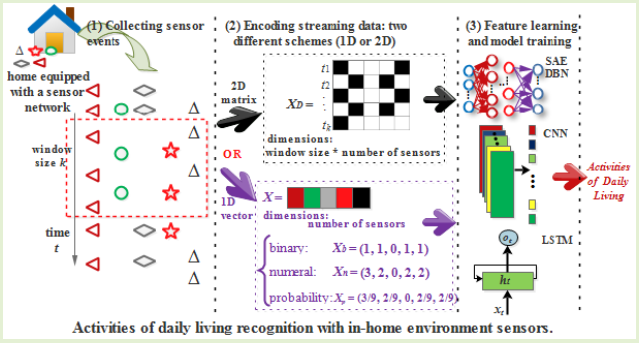
Activities of Daily Living Recognition With Binary Environment Sensors Using Deep Learning: A Comparative Study

Aiguo Wang, Shenghui Zhao, Chundi Zheng, Jing Yang, Guilin Chen, and Chih-Yung Chang, *Member, IEEE*

***Abstract*—The power of end-to-end deep learning techniques to automatically learn latent high-level features from raw signals has been demonstrated in numerous application fields, however, few studies systematically investigate how to properly encode the time-series firings of binary environment sensors that typically work in an event-triggering scheme and have irregular sampling rates for in-home human activity recognition. To this end, we here propose two different methods to process the streaming sensor readings and accordingly evaluate their combinations with deep learning models. Specifically, we divide the multichannel sensor events into segments and encode each segment into either a vector or two-dimensional matrix. Particularly, three different feature representations are presented for the vector form. Afterwards, we combine the encoded features with four typical deep learning models and optimize corresponding activity recognizers to study their sensitivity to different feature encodings. Furthermore, we include seven commonly used shallow classification models for comparison purposes. Finally, we conduct extensive experiments on three publicly available smart home datasets. Results indicate that the performance of both deep learning and shallow models is closely associated with the raw signal encodings and demonstrate the superiority of one-dimensional convolutional neural networks over its competitors in terms of generalization across scenarios. Besides, we preliminarily explore the influence of NULL class on an activity recognizer and experimentally show its negative impact on overall accuracy, enlightening relevant studies to consider it in developing a practical activity recognition system.**



Activities of daily living recognition with in-home environment sensors.

***Index Terms*—Feature encoding, feature learning, activity recognition, smart home.**

I. Introduction

IT HAS been known that our world has been stepping towards the aging society and facing significant economic and social issues such as the financial stress, increasing elderly healthcare demands, and unbalanced supply-demand. Consequently, due to the high expenditure of healthcare costs and the willingness of the elderly to live independently in their own places, smart homes equipped with various types of sensors and actuators are consistently emerging with an aim to provide pervasive and context-aware services and to help maintain a healthy, safe, and functional life [1], [2]. Particularly, with the development of sensor technology, pervasive computing, internet of things, and artificial intelligence, ambient assisted living systems deployed in smart homes can intelligently perceive and act on physical surroundings and potentially support a variety of applications that range from fall detection, long-term behavior pattern analyses, and health and wellness evaluation to chronic disease management, consistent rehabilitation instruction, and timely medication reminder [3]–[5]. For residents wish to stay at their own places independently and functionally, they should have the ability to conduct Activities of Daily Living (ADLs) such as cooking, drinking, preparing dinner, washing grooming, and eating [6], and healthcare providers use ADLs to measure an individual's functional status. Therefore, accurately recognizing in-home

Manuscript received October 22, 2020; revised October 27, 2020; accepted October 28, 2020. Date of publication October 30, 2020; date of current version January 15, 2021. This work was supported in part by the Natural Science Foundation of China under Grant 61902068 and Grant 61972092, in part by the Major Special Projects of Anhui Province under Grant 201903A06020026, in part by the Key Research and Development Project of Anhui Province under Grant KJ2019ZD44, and in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2020A1515011499. The associate editor coordinating the review of this article and approving it for publication was Prof. Elena Gaura. *(Corresponding author: Guilin Chen.)*

Aiguo Wang and Chundi Zheng are with the School of Electronic Information Engineering, Foshan University, Foshan 528225, China (e-mail: wangaiguo2546@163.com; cdzheng@fosu.edu.cn).

Shenghui Zhao and Guilin Chen are with the School of Computer and Information Engineering, Chuzhou University, Chuzhou 239000, China (e-mail: zsh@chzu.edu.cn; glchen@chzu.edu.cn).

Jing Yang is with the School of Computer and Information, Hefei University of Technology, Hefei 230009, China (e-mail: jsjyj0801@163.com).

Chih-Yung Chang is with the Department of Computer Science and Information Engineering, Tamkang University, New Taipei City 25137, Taiwan (e-mail: cychang@mail.tku.edu.tw).

Digital Object Identifier 10.1109/JSEN.2020.3035062

activities plays an essential role in understanding the relationship between residents and their surroundings, where it functions as middleware in bridging the low-level sensor data and high-level human-centric applications [7], [8].

Due to the inherent complexity of human behavior that is characterized by uncertainty and diversity, automatically and accurately inferring ongoing activities remains a challenging yet rewarding research topic that has been attracting attentions from researchers [9]. Typically, there exist *inter-subject variation*, where different people can perform the same activity in a different way, and *intra-subject variation*, where one performs an activity differently in different situations [10]. For example, an individual probably prepares dinner in a different way of using kitchenware and dinnerware, and one walks with varying step sizes and speeds at different time. Also, there are activities that can trigger similar sensor readings, which deteriorates the performance of an activity recognizer [10]. With the aim to adapt to various application scenarios and to obtain satisfactory recognition performance, researchers have used different kinds of sensing technologies and models towards adaptive activity recognition. Generally, existing activity recognition methods could be broadly grouped into three categories based on the used sensing technology [8], [11]: vision-, wearable sensor-, and environment sensor-based methods. Vision methods capture a series of images with a camera or video to recognize activities and they have a wide range of applications such as surveillance, security and safety [12]. One typical example is the use of depth-camera for motion sensing games. One major drawback is that they suffer from the privacy issue and are not welcome in a smart home setting [13]. Their actual use is further limited by ambient occlusion, illumination variations, and environmental noise [12]. Wearable sensor-based methods benefit from the miniature of sensing units and they recognize human activities by first collecting sensor data from wearable devices and then training an activity recognizer [14]. Particularly, we get multiple heterogeneous or homogeneous sensing units attached to different parts of human body (e.g., the wrist, arm, leg, and waist) [15]. Obviously, one is required to carry/wear one or more devices all the time, which inevitably brings inconvenience to individuals when performing ADLs [16]. This largely prevents it from being an ideal plan for the elderly care. In contrast, environment sensor-based methods place sensing units on household objects and infer ADLs by capturing and analyzing the interaction between an individual and the ambient objects [2], [17]. For example, Yatbaz *et al.* used a set of simple state-change sensors to refer human activities [16]. Cook *et al.* implemented the CASAS architecture that utilized contact sensors, motion sensors and infrared detectors to support high-level applications [2]. Such methods have advantages of easy deployment, low costs, and inherent less-intrusiveness, and thus are considered a promising way to automate the recognition of in-home ADLs.

According to the activity recognition chain that consists of data collection, feature encoding, model optimization and prediction, the performance of an activity recognizer is largely determined by the choice of classification models and the way to encode the sensor data [10], [18]. Accordingly, researchers have conducted considerable work in exploring effective methods that range from generative models (e.g., naïve Bayes and hidden Markov model) to discriminant models (e.g., support vector machine, decision tree, and conditional random field) [13]. Since most of them have a shallow structure and rely on hand-crafted features, they probably fail to capture the complex relationships among the raw sensor signals. In contrast, deep learning models have the end-to-end learning capability to automatically learn high-level features from raw signals without the guidance of human experts, which facilitates their wide applications in fields such as speech recognition, computer vision, and natural language processing [19]. They have also advanced wearable sensor-based activity recognition. However, different from wearable sensors that generate sensor readings with constant sampling rates, environment sensors work in an event-triggering scheme and have irregular sampling rates. How to encode the events of environment sensors remains underexplored and this presents a challenge to the proper use of multichannel time-series sensor events for inferring ADLs. Although there are many classification models available for activity recognition, few studies, to the best of our knowledge, systematically investigate how to effectively encode the binary streaming events and accordingly evaluate the relationships with the finally optimized activity recognizer. Besides, different from the traditional classification problems, activity recognition inherently faces the NULL class problem, where parts of the sensor readings are irrelevant to the predefined activities of an application. In most cases, the NULL class represents a large unknown space and is accompanied with human behavior and it is difficult, if not impossible, to enumerate all activities in a specific application, however, few studies consider it and further quantitatively evaluate its influence on an activity recognizer. To this end, we herein present two different methods to process the time-series sensor events and evaluate their combinations with deep learning and shallow models and further evaluate the NULL class problem. This potentially guides users in handling the streaming environment sensor signals and enlightens further studies to consider the NULL class in developing real-world applications. The main contributions of this study are itemized as follows. (1) We analyze and compare the signals of wearable sensors and environment sensors and propose two methods to handle the multichannel binary sensor data. Specifically, we encode the sensor data into either a vector or two-dimensional matrix. Furthermore, we present three specific instantiations for the vector form, including *binary*, *numeral*, and *probability representations*. This guides users in how to encode the sensor events. (2) We detail how to combine the encoded features to four typical deep learning models (i.e., autoencoder, deep belief network, convolutional neural network, and long-short term memory network). Moreover, we include other seven shallow models for comparison purposes. (3) Due to characteristics of human behavior, it is not trivial to filter out the NULL class if an individual is allowed to perform activities in a natural setting. We explicitly evaluate the influence of NULL class on an activity recognizer with the aim to prompt users to consider it in comprehensively evaluating and developing a practical activity recognizer. (4) Extensive experiments are

conducted on public smart home datasets collected with binary environment sensors. Results indicate that the performance of both deep learning and shallow models is closely associated with the feature encodings and that the NULL class generally lowers recognition accuracy.

The rest of this paper is organized as follows. In section II, we review related work on human activity recognition from the perspective of prediction models. Section III illustrates how to encode streaming binary sensor data and details how to feed the encoded features to deep learning models. Environmental setup, results and analyses are illustrated in section IV. The last section concludes this study with a brief summary and discussions.

II. Related Work

To adapt to various human-centric application scenarios and to obtain satisfactory recognition performance, recent years have witnessed considerable work in developing a wealth of sensing techniques and a number of models for environment sensor-based activity recognition [20], [21]. We can broadly group existing activity recognizers into knowledge-driven methods and data-driven methods [22], [23]. The former requires an abstract model of domain knowledge to define activity specification. They have an advantage of easy interpretation and being robust to noise and incomplete data [24]. For example, Chen *et al.* used the logical reasoning and logical knowledge to explicitly model context and activities [25]. One limitation of such methods is the acquisition of expert knowledge of a specific domain. In contrast, data-driven methods only rely on data to train an activity recognizer that associates sensor data with corresponding activity labels. Accordingly, researchers have presented many models that range from generative models (e.g., naïve Bayes and hidden Markov model) and discriminant models (e.g., support vector machine and decision tree) to ensemble models (e.g., bagging, boosting and stacking) [10]. For example, Ordónez *et al.* combined artificial neural networks and support vector machine within the hidden Markov model to train an activity recognizer [26]. However, due to the inherent complexity of human behavior, shallow models probably fail to learn the non-linear relationships among raw sensor data [18]. Moreover, most of them rely on domain knowledge to extract features and treat feature extraction and classifier training as two separate steps, which leads to sub-optimization.

In contrast to shallow models, deep learning models have the ability to automatically learn complex features from low-level signals and jointly optimize the steps of feature learning and classifier training. There are studies that apply deep learning models to activity recognition [27]. For example, Plötz *et al.* are among the first researchers that used principal component analysis and deep belief networks (DBN) to develop an activity recognition framework for wearable computing applications [22]. They conducted experiments on accelerometer data, which showed the benefits of feature learning. Particularly, DBN nonlinearly transforms the original features into high-level features. Teng *et al.* applied deep convolution neural networks (CNNs) on time-series data for activity recognition [28]. Experimental results indicate the proposed model outperforms its competitors. Ronao and Cho applied a deep CNN to the data collected with smartphones and trained an effective and efficient activity recognition model [29]. To use the temporal dynamics associated with activities, Ordóñez and Roggen proposed a convolutional long-short term memory (LSTM) recurrent neural network and applied it to automatically learn latent features from multimodal wearable sensor data and to train an activity recognizer [30]. They evaluated the model on two datasets and experimental results showed its effectiveness. Guan *et al.* combined diverse LSTMs under an ensemble framework for wearable sensor-based activity recognition [31]. Experimental results showed the superiority of the LSTM ensemble over its single component. Besides wearable sensors, researcher have applied deep learning to train environment sensor-based activity recognition models. For example, Chen *et al.* used a denoising autoencoder network to train an activity recognizer and compared it with DBN [7]. Gochoo *et al.* used deep CNNs to recognize ADLs of the elderly [30], where they took as input the activity image and applied the two-dimensional convolution to learn features. Liciotti *et al.* explored LSTM networks and proposed several variants for modelling the temporal sequences of sensor events [32]. They fed the sensor events into LSTM-based model. The above work advances the in-home activity recognition with environment sensors, while few studies systematically investigate how to properly encode the streaming binary events and accordingly evaluate their combinations with deep learning models. Particularly, although deep learning models trained on wearable sensor data provide users valuable experience, the two types of sensors have inherent differences. Wearable sensors usually work at constant sampling rates and output continuous sensor readings, and the signal intensities are naturally used to infer on-going human activities. In contrast, environment sensors, working in an event-triggering scheme, only generate firings when there are interactions between an individual and the ambient objects, and thus they have irregular sampling rates. This presents a new scenario for streaming data analysis and also motivates us to study how to encode the multichannel streaming binary events for activity recognition. We here explore two different schemes to encode sensor events into either a vector or two-dimensional matrix. Furthermore, three different feature representations are given for the vector form. Afterwards, we evaluate their combinations with deep learning models and also study the sensitivity of shallow models to different feature encodings. In addition, most studies only consider the activities of interest and ignore the NULL class in training activity recognizers. However, it is inapplicable to real-world applications, since the NULL class is inherently accompanied with human behavior. Accordingly, we conduct a comparative study on the influence of NULL class to an activity recognizer with an aim to enlighten relevant studies to consider it in developing a practical activity recognition system.

III. Activity Recognition With Environment Sensors

In-home activity recognition aims to automate the recognition of activities of daily living using a sensing system

equipped with simple state-change environment sensors. It mainly includes three stages to return an activity recognizer. First, the streaming sensor events are transmitted to a central processing platform through a wireless network and the data is divided into segments through the sliding window techniques. Second, we encode the streaming events, return feature vectors for each segment, and optimize an activity recognizer. Finally, we use the trained model to infer the activity label of a stream of sensor events. In the training stage, we need to collect sensor data and annotate them with corresponding activity labels. Afterwards, we extract features from the segment to optimize a classifier. Considering that how to encode the sensor events largely determines the performance of an activity recognizer, we explore different schemes to encode the multichannel streaming sensor events and use deep learning techniques to jointly optimize the feature learning and classifier training towards better learning complex relationships among data. After obtaining an activity recognizer, we use it to predict the activity label of unseen sensor events.

A. Encoding Streaming Sensor Events

Due to the inherent complexity of human behavior, inferring on-going activities according to the sensor readings of a single time point is quite difficult and one feasible way is to segment the time-series data into chunks. In processing streaming data, sliding-window techniques have been commonly used and three specific methods are available for use: explicit segmentation, sensor event-based windowing, and time-based windowing [23]. For explicit segmentation, it perfectly relates an activity to corresponding sensor events and can get an activity recognizer that works well on the pre-segmented sequences. However, such a method is inappropriate to real-world applications, because it is difficult to determine the occurrence of a performed activity without human supervision. In contrast, time-based windowing method divides the sequence into time intervals of equal size. Compared with the pre-segmentation scheme, it provides a practical solution and has been commonly used in handling time-series data. Similarly, sensor event-based windowing method divides a sensor event sequence into chunks of equal number of events. Since different activities probably trigger a set of different sensors at different times and in different places, it can generate chunks of different durations. Particularly, both time-based and sensor event-based windowing methods require users to specify the window size. A small-size window would fail to contain discriminant information of an activity, while a large-size window may involve the sensor readings of multiple activities. In addition, according to whether the window size is fixed, sliding window techniques are further divided into two schemes: fixed length and variable length.

In this study, we use the time-based sliding window with a fixed size to analyze the time-series sensor events and prepare them for subsequent classification models. Fig. 1 presents an illustrative example on applying a sliding window of size k to the multichannel streaming sensor events, where a window shifts over time with a user-defined time step and the horizontal axis means different sensors. Obviously, the returned segment forms a two-dimensional (2D) matrix, with columns

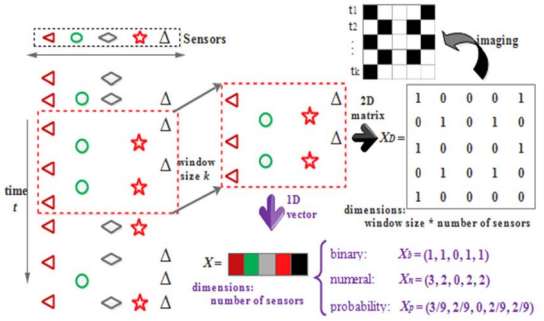


Fig. 1. Illustration of how to code the multichannel streaming binary sensor events. A sliding window of size k shifts over the time to get segments, from which we obtain the 2D and 1D feature encodings.

indicating different environment sensors and rows denoting time steps. Its element takes a value of 1 if corresponding sensor is triggered at that time slot; otherwise, 0. We further aggregate the segment to form a one-dimensional (1D) vector with length N, where N is the number of sensors deployed in a smart home. For the vector form, we investigate three different ways to code the segment: *binary representation*, *numeral representation*, and *probability representation*. The numerical representation counts the number of firings of a sensor in a segment, probability representation denotes the ratio of firings of a sensor, and binary representation shows whether a sensor reported an event. For the case in Fig. 1, given five sensors and a window of depth five, if the first sensor fired three times, the third sensor was not triggered, and the rest sensors all fired twice in the time slice, feature vectors of the three representations are $X_b = (1, 1, 0, 1, 1)$, $X_n = (3, 2, 0, 2, 2)$, and $X_p = (3/9, 2/9, 0, 2/9, 2/9)$, respectively. After detailing how to encode the multichannel streaming events, we show how to combine them with deep learning models.

B. Autoencoder

An autoencoder, typically consists of one input layer, at least one hidden layer, and one output layer, aims to reconstruct its input in the output layer with minimal errors. Specifically, for a N-dimensional $x = (x_1, x_2, \ldots, x_N)$, an autoencoder first encodes it into a M-dimensional latent vector $h = (h_1, h_2, \ldots, h_M)$ using (1),

$$h(x) = f(W^{(1)}x + b^{(1)}), \tag{1}$$

where $W^{(1)} \in \mathrm{R}^{N\times M}$ is the weight matrix between the input and hidden layers, $b^{(1)}$ stores the biases of h, and f denotes an activation function. We reconstruct x from $h(x)$ using (2) with an objective of minimizing the difference between x and x_{est}.

$$x_{est} = f(W^{(2)}h(x) + b^{(2)}), \tag{2}$$

where $W^{(2)} \in \mathrm{R}^{M\times N}$ is the weight matrix between the output layer and hidden layer and $b^{(2)}$ stores the biases.

Furthermore, previous studies show that a deep architecture helps discover the highly non-linear relationships among data. A stacked autoencoder (SAE) takes an autoencoder as the building block and is trained using the greedy layer-wise

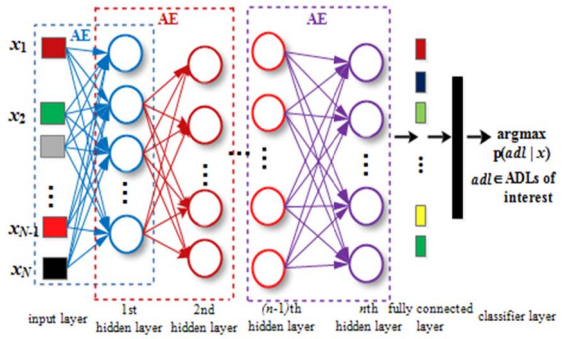


Fig. 2. SAE-based activity recognition model with environment sensors.

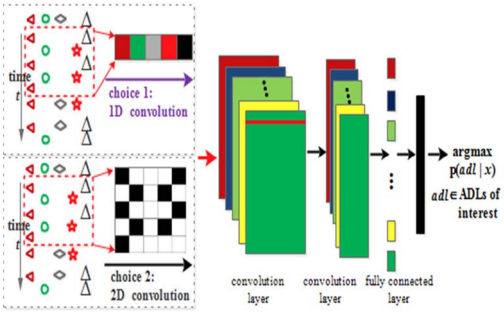


Fig. 3. CNN-based activity recognition model with binary environment sensors. It can take as input either the 1D vector (shown in the upper left part) or 2D matrix (shown in the lower left part).

learning. Given a stacked autoencoder with n layers, for the p-th autoencoder ($1 \leq p \leq n$), we perform (3) and (4) iteratively,

$$a^{(p)} = f(z^{(p)}) \tag{3}$$

$$z^{(p+1)}) = W^{(p)}a^{(p)} + b^{(p)}, \tag{4}$$

where $z^{(p)}$ is the input of the p-th layer, $a^{(p)}$ is its activation, and $a^{(1)} = x$ when $p = 1$. Finally, we stack a fully connected layer and a classification layer on top of the SAE and fine-tune the network parameters [18]. Fig. 2 presents SAE-based activity recognition model.

C. Deep Belief Network

Similar to SAE, deep belief network (DBN) is a stack model with restricted Boltzman machines (RBM), where the hidden layer of each previous sub-network is the visible units of subsequent layer [18]. An RBM is a generative energy-based model that consists of a visible layer and a hidden layer and it is trained by contrastive divergence. For a N-dimensional visible vector $v = (v_1, v_2, \ldots, v_N)$, RBM updates the hidden units with the visible units based on (5),

$$p(h_j = 1|\mathbf{V}) = sigmod(\sum_i v_i w_{ij} + b_j), \tag{5}$$

then updates the visible units with the hidden units using (6),

$$p(v_j = 1|\mathbf{H}) = sigmod(\sum_j h_j w_{ij} + a_i), \tag{6}$$

where a and b are the biases of the visible layer and hidden layer, respectively. RBM re-updates the hidden layer with the reconstructed visible layer and updates the weights W.

After training an RBM, we stack another RBM on top of it and take the hidden layer as the visible layer of next RBM. A greedy layer-wise scheme is used to train a DBN. Finally, we stack a fully connected layer and a classification layer on top of the DBN and fine-tune the network parameters on the basis of the pretrained network. That is, DBN has a structure similar to SAE, except the RBM building block.

D. Convolutional Neural Network

Convolutional neural network (CNN) generally comprises an input layer, at least one convolution and pooling layer, and at least one fully-connected layer [28]. Compared with SAE and DBN, CNN, having the characteristics of weight sharing and translation invariance, can better capture the local dependencies in lower layers and salient patterns in higher layers. Since each layer typically contains a number of convolution operators. This enables CNN to get multiple salient patterns that are learned from different views. Specifically, given the previous layer of CNN, we use (7) to obtain the feature map of the next layer.

$$a_j^{(l+1)}(\tau) = \sigma(b_j^l + \sum_{i=1}^{F_l}[\sum_{k=1}^{K_w} K_{ji}^l(k)a_f^l(\tau - k)]), \tag{7}$$

where $a_f^l(\tau)$ means the feature map j of layer l, F_l denotes the number of feature maps of layer l, K_{ji}^l is the kernel that is used to get the feature map j of layer $(l+1)$, K_w is the kernel width, and σ is an activation function. Finally, we stack a fully-connected layer and a classification layer to train an activity recognition model. Significantly, the convolution kernel depends on the input dimensionality. Herein, we explore both 1D vector and 2D matrix feature encodings, where we apply a 1D convolutional kernel on the 1D input and 2D kernel on the 2D input. Fig. 3 presents the corresponding CNN architectures.

E. Long-Short Term Memory Network

Time-series data generally contain temporal information to reflect the dependencies, and long-short term memory (LSTM) network is designed to make use of such information. LSTM uses the gating mechanism to learn the temporal dynamics and updates the cell state using input gate, output gate, and forget gate. Specifically, given a temporal input sequence $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$, it performs the following steps to calculate the hidden state h_t $(1 \leq t \leq T)$,

$$\mathbf{i}_t = \sigma_i(W_{xi}\mathbf{x}_t + W_{hi}\mathbf{h}_{t-1} + W_{ci}\mathbf{c}_{t-1} + \mathbf{b}_i) \tag{8}$$

$$\mathbf{f}_t = \sigma_f(W_{xf}\mathbf{x}_t + W_{hf}\mathbf{h}_{t-1} + W_{cf}\mathbf{c}_{t-1} + \mathbf{b}_f) \tag{9}$$

$$\mathbf{c}_t = \mathbf{f}_t * \mathbf{c}_{t-1} + \mathbf{i}_t * \sigma_c(W_{xc}\mathbf{x}_t + W_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c) \tag{10}$$

$$\mathbf{o}_t = \sigma_o(W_{xo}\mathbf{x}_t + W_{ho}\mathbf{h}_{t-1} + W_{co}\mathbf{c}_{t-1} + \mathbf{b}_o) \tag{11}$$

$$\mathbf{h}_t = \mathbf{o}_t * \sigma_h(\mathbf{c}_t), \tag{12}$$

where $\mathbf{i}$, $\mathbf{o}$, $\mathbf{f}$, and $\mathbf{c}$ represent the input gate, output gate, forget gate, and cell activation, respectively, σ denotes an activation function, $\mathbf{x}_t$ denotes input at time t, W are the

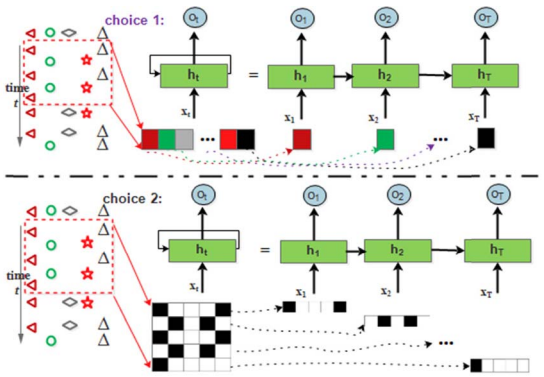


Fig. 4. LSTM-based activity recognition models. The upper part presents the LSTM for 1D input and the lower part corresponds to 2D case.

associated weight matrices, and **b** indicates the bias vector. We present two LSTM networks corresponding to different feature encodings, shown in Fig. 4, where the upper part illustrates how to customize the LSTM for 1D feature space and the lower part corresponds to 2D feature space. Their major difference is their input. For 1D representation, its input $\mathbf{x}_t$ is a scalar and the sequence length equals the number of sensors, while the sequence length equals the width of sliding window and its input $\mathbf{x}_t$ is a N-dimensional vector for 2D representation.

IV. Experimental Setting and Results

A. Sensing Network

To evaluate different original feature encoding schemes and their combinations with deep learning models, we conduct comparative experiments on three smart home datasets that are collected with binary environment sensors [13]. Specifically, to automate the recognition of ADLs, different types of sensors are attached to or embedded into ambient objects in homes: pressure mats to measure lying in bed or sitting on a chair, passive infrared to detect motion in a specific area, float sensors to measure the use of the toilet, mercury contacts to detect the movement of objects, and reed switches to indicate the status of door or cupboard open-close. A sensor network consisting of a collection of these sensors observes and records the behavior of the residents. The network works with an energy-saving protocol and a 4.8kb/s data transmission rate, which is enough for long-term data collection, and it also has great scalability in dynamically adding or removing sensing units. Besides, the sensor network is easily deployed using some tape at low cost and less intrusive to residents compared with wearable devices because sensors are mounted on or embedded into objects. If the state of an object changes, the associated sensors generate binary outputs and send them wirelessly to remote servers for further analysis. Afterwards, sensor events are collected from the smart homes and annotated with corresponding activities via online and/or offline annotation methods (e.g., time diary, audio recordings, experience sampling, and self-recall) when an individual performs activities inside the home. A Bluetooth headset is used to annotate the start and end time of activities. Moreover, the same clock is used to timestamp sensor events towards well-synchronized measurements.

TABLE I
Description of the Three Smart Home Datasets

Smart home	SH-a	SH-b	SH-c
#resident	1	1	1
resident age	26	28	57
#rooms	3	2	6
#days monitored	25	14	19
#sensors	14	23	21
#activities	10	13	16
#sensor events	1229	19,075	22,700
#activity instances	292	200	344

B. Experimental Data

Since it is difficult, if not impossible, to consider all activities due to the complexity of human behavior and the level in defining an activity (e.g., unit-level action and composite activity), in this study, the activities of interest are chosen based on the Katz ADL index that assesses the elderly physical and cognitive capabilities and they are a subset of both basic ADLs and instrumental ADLs. Basic ADLs are mainly related to self-care tasks in maintaining one's fundamental functioning, and the instrumental ADLs include the activities that enables one to live independently in a community. The first smart home (SH-a) installed with fourteen sensors has three rooms and houses an independent living resident. SH-a collects sensor data for twenty-five days, resulting in 292 activity instances and 1229 sensor events. Besides, we include NULL class, which refers to times at which no human activity is annotated. It may contain background data and activities that are considered by an application. The second smart home (SH-b) is an apartment and twenty-three sensors are deployed in different places such as the kitchen, bathroom, bedroom, dining room, and living room. During the data collection stage, a volunteer stays inside it for two weeks. The SH-b dataset has 200 activity instances. In the experiment, sensor data associated with twelve predefined activities are annotated. Different from SH-a and SH-b, the third smart home (SH-c) is a house with two floors and installed with twenty-one sensors. Experimental data are continually collected for nineteen days when a senior person stays independently in the house, and sixteen ADLs are considered. This makes the recognition task much more challenging than that of SH-a and SH-b. Table I presents the description of the three smart homes, and readers can refer to [13] for details about experimental protocols, sensor types, and data annotation.

C. Experimental Setup

As for the feature encoding, we herein apply a time-based sliding window to divide time-series sensor data into segments. According to previous studies, a window size of suggested 60 seconds is used to shift over the multichannel streaming sensor data [7], [30]. Particularly, as detailed previously, we present two different original feature encodings (i.e., 1D vector and 2D matrix) and give three specific forms for 1D vector (i.e.,

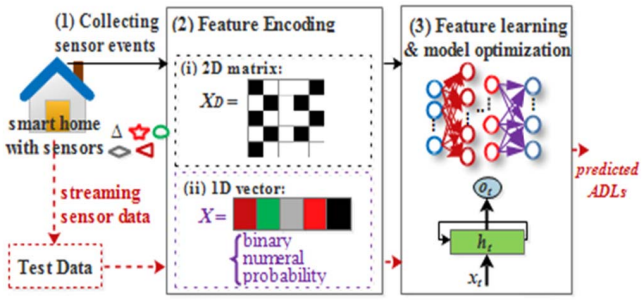


Fig. 5. The proposed in-home activity recognition model.

TABLE II
EXPERIMENTAL SETUP

Model	Architecture	Parameter and values
SAE	*A-A-A-FC-Sm*	degree of sparsity: 0.15, weight regularization: 0.004, activation: Sigm, optimizer: CG
DBN	*R-R-R-FC-Sm*	activation: Sigm, optimizer: Adam (0.001)
CNN1d	*C-C-C-FC-Sm*	dropout: 0.5, kernel: 1*2/1*3, activation: ReLU, optimizer: Adam (0.001)
CNN	*C-C-C-FC-Sm*	dropout: 0.5, kernel: 2*2/3*3, activation: ReLU, optimizer: Adam (0.001)
LSTM1d	*L-L-FC-Sm*	activation: Tanh, optimizer: Adam (0.001)
LSTM	*L-L-FC-Sm*	dropout: 0.5, activation: Tanh, optimizer: Adam (0.001)

binary, *numeral*, and *probability representations*). Afterwards, we feed the encoded features to classification models to train an activity recognizer. To evaluate the power of a classifier, we use the leave one day out cross validation, where one full day of sensor data is used as the test set and data from the remaining days are used to train a classifier. We repeat the above process the total number of days times such that each day will be the test set and we report the average classification accuracy. Fig. 5 illustrates the experimental setup that consists of the training stage shown by the black solid line and the test stage indicated by the red dashed line. In the training stage, we collect the sensor events and optimize an activity recognizer, which we use to predict the activity labels of the test data.

As for the choice of classification models, we explore four commonly used deep learning models (SAE, DBN, CNN, and LSTM) and comparatively evaluate their responses to different feature encodings. Specifically, we build a three-layer stacked autoencoder and consider different number of units in a hidden layer (i.e., $M = 8$, 16, or 32). We stack a Softmax layer (*Sm*) on top of SAE, and the number of hidden units of *Sm* equals the number of activities of interest. To optimize the parameters of SAE, we use a greedy layer-wise way to train it with conjugate gradient descent (CG) and do fine-tuning in an up-bottom way. For DBN, we also explore a three-layer architecture towards a relatively fair comparison to SAE. We use the pretrain strategy to build DBN, stack a Softmax layer, and fine-tune it using the available labeled data. We consider different number of units in a hidden layer, i.e., 8, 16, and 32. Since SAE and DBN typically work with vectors, their input takes a 1D vector and both have three different encodings. For CNN, it has the scale invariance and local dependency and potentially learns hierarchical representations from raw sensor data. Inspired by the structure of LeNet-5, we use three convolution layers, followed by a fully connected layer with a 0.5 dropout rate and a Softmax layer. Particularly, we consider two specific CNNs: 1D CNN and 2D CNN. The former works on 1D vector sequences and utilizes 1D convolution kernel to learn local features, while the latter takes as input 2D sequences and uses 2D convolution kernel to learn high-level features. We use the rectified linear unit (ReLU) activation function in convolution layers and set the number of hidden units of the fully connected layers to be the number of predefined activities. We consider different number of feature maps in a hidden layer (i.e., 8, 16, and 32) and explore different kernel sizes. The evaluated kernel sizes are 2 and 3. Besides, the two CNNs are optimized by the stochastic gradient descent with momentum and a learning rate of 0.001 (Adam). For LSTM, according to the empirical studies of [19] suggesting that a two-layer LSTM performs better, we include two-layer LSTM networks and stack a fully connected layer and a Softmax layer. We here present two specific LSTMs: 1D LSTM and 2D LSTM. The former works on 1D sequences and its time step is one, while the latter takes as input 2D sequences and its time step is the observation over time. The activation function of LSTM is the hyperbolic tangent function. We also consider different number of hidden units of a LSTM layer, and the candidate values are 8, 16, and 32. For 2D LSTM, dropout is used in the first two hidden layers with a 0.5 probability. Both LSTMs are optimized using Adam with a 0.001 learning rate. Table II presents the experimental setup, where *A* indicates an autoencoder, *R* means an RBM, *C* denotes a convolutional layer, *L* refers to a LSTM layer, *FC* denotes a fully connected layer, and *Sm* is the Softmax layer. The size of input vector depends on the feature encodings, and *Sm* has the same number of nodes as the number of predefined activities.

In addition, we include other seven commonly used models with shallow structures, including naïve Bayes (NB), hidden Markov model (HMM), hidden semi-Markov model (HSMM), k-nearest-neighbor (KNN), support vector machine with linear kernel (SVM), multilayer perception (MLP), and decision tree (C4.5), to not only compare them with deep learning based activity recognizers, but also to evaluate their sensitivity to different feature representations. For shallow models, we take as input the 1D original features. Default parameter values are used for SVM and the nearest neighbor is used to predict the labels of unseen samples in KNN.

D. Experimental Results Without NULL Class

We first conduct experiments in the situation where only predefined activities are considered. As for the choice of feature encodings, besides CNN and LSTM that handle both 1D and 2D features, the rest use 1D input. Tables III-V presents the results on the three datasets, respectively. The first column "Feature" denotes the different feature representations: *binary* (Binary), *numeral* (Numeral), and *probability representations* (Probability). As 2D representation consists of binary values,

TABLE III
EXPERIMENTAL RESULTS ON HOUSE A WITHOUT NULL CLASS

Feature	NB	HMM	HSMM	1NN	SVM	MLP	C4.5	SAE	DBN	CNN1d	CNN	LSTM1d	LSTM
Binary	86.53	61.99	67.51	37.92	93.69	93.67	93.71	90.43	94.13	**94.51**	93.49	93.40	93.32
Numeral	86.34	61.71	73.38	38.30	93.74	94.28	93.90	88.28	62.03	**94.83**	-	93.92	-
Probability	83.36	43.51	59.80	38.28	85.46	93.37	**93.72**	60.46	61.44	93.60	-	92.98	-

TABLE IV
EXPERIMENTAL RESULTS ON HOUSE B WITHOUT NULL CLASS

Feature	NB	HMM	HSMM	1NN	SVM	MLP	C4.5	SAE	DBN	CNN1d	CNN	LSTM1d	LSTM
Binary	89.17	62.24	65.53	64.90	81.45	85.49	81.70	**96.31**	88.54	95.43	87.51	91.15	87.97
Numeral	88.44	71.41	71.54	71.35	86.31	96.59	84.33	82.45	64.80	**96.60**	-	94.73	-
Probability	68.62	89.05	89.05	70.98	92.53	**96.25**	80.38	63.30	61.95	94.99	-	63.44	-

TABLE V
EXPERIMENTAL RESULTS ON HOUSE C WITHOUT NULL CLASS

Feature	NB	HMM	HSMM	1NN	SVM	MLP	C4.5	SAE	DBN	CNN1d	CNN	LSTM1d	LSTM
Binary	49.17	24.79	27.00	26.91	54.21	39.53	34.21	53.29	**57.96**	57.88	44.70	53.29	53.72
Numeral	47.35	27.04	38.53	33.06	52.88	52.91	40.91	60.45	53.72	**61.55**	-	58.73	-
Probability	49.88	34.53	40.03	34.03	**61.18**	48.16	43.21	53.04	48.22	56.58	-	53.04	-

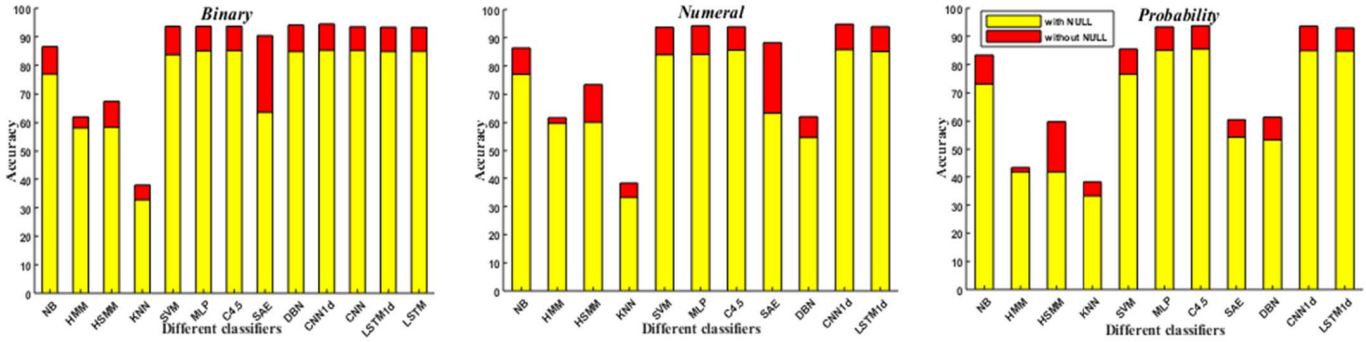


Fig. 6. Performance comparison between with and without NULL class on SH-a.

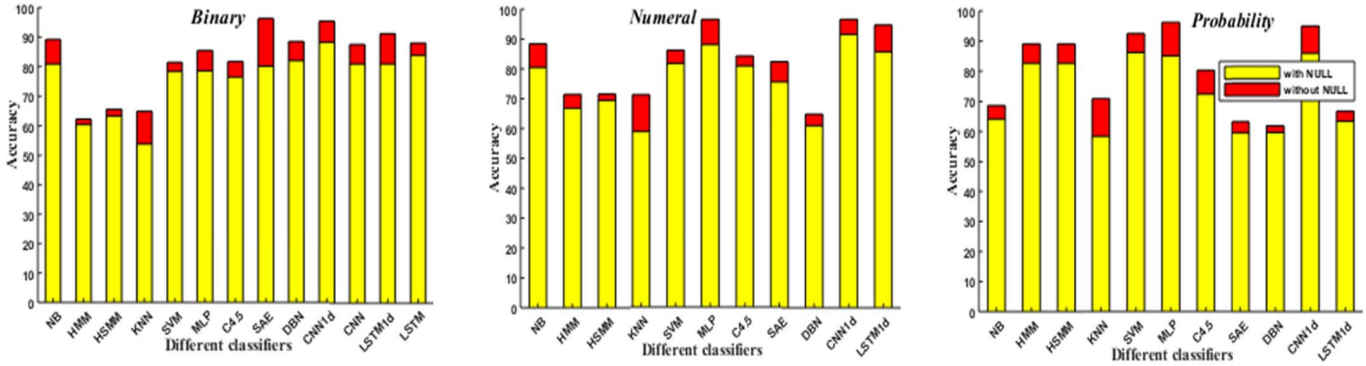


Fig. 7. Performance comparison between with and without NULL class on SH-b.

we categorize its results into Binary. CNN1d and LSTM1d denote results with 1D input, and CNN and LSTM shows the results with 2D input. The best result on each representation is shown in bold and the best result on each dataset is underlined.

From Table III, we observe that one classifier, even the deep learning model, generally obtains different performance across original feature encodings. For example, NB achieves accuracy of 86.53%, 86.34%, and 83.36% with the three representations, respectively. DBN gets 94.13% accuracy with the binary representation and 61.44% accuracy with the probability representation. SAE achieves accuracy of 90.43%, 88.28%, and 60.46% with binary, numeral, and probability representations, respectively. We see CNN1d and LSTM1d obtain relatively stable results compared with DBN and SAE.

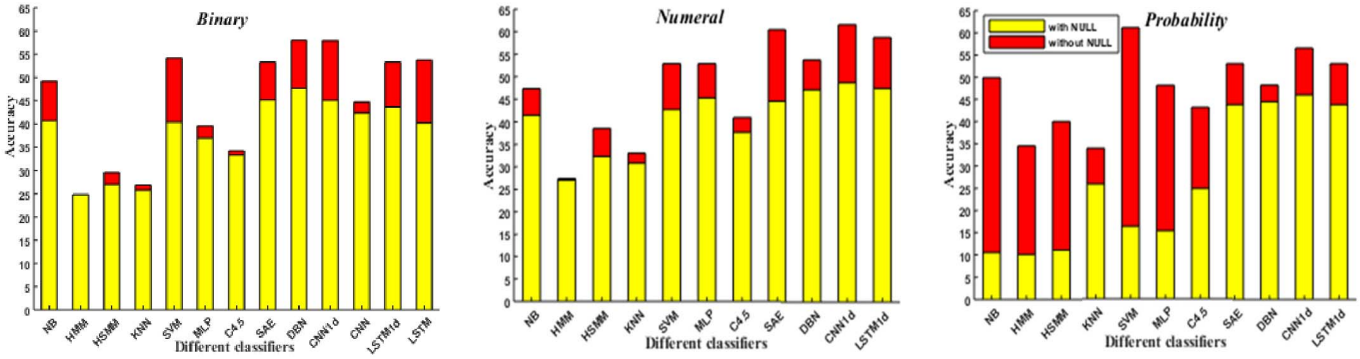


Fig. 8. Performance comparison between with and without NULL class on SH-c.

TABLE VI
EXPERIMENTAL RESULTS ON HOUSE A WITH NULL CLASS

Feature	NB	HMM	HSMM	1NN	SVM	MLP	C4.5	SAE	DBN	CNN1d	CNN	LSTM1d	LSTM
Binary	76.90	58.13	58.39	32.78	83.70	85.05	85.16	63.65	84.87	**85.30**	85.23	84.85	84.88
Numeral	77.03	59.73	60.10	33.30	83.95	84.12	85.59	63.35	54.65	**85.79**	-	85.13	-
Probability	73.03	41.77	41.77	33.29	76.56	**85.06**	85.49	54.28	53.35	84.88	-	84.80	-

TABLE VII
EXPERIMENTAL RESULTS ON HOUSE B WITH NULL CLASS

Feature	NB	HMM	HSMM	1NN	SVM	MLP	C4.5	SAE	DBN	CNN1d	CNN	LSTM1d	LSTM
Binary	80.88	60.49	63.22	53.84	78.40	78.62	76.46	80.15	82.09	**88.25**	80.88	80.89	83.85
Numeral	80.50	66.79	69.42	59.03	81.82	88.15	81.02	75.68	60.94	**91.54**	-	85.73	-
Probability	64.22	82.64	82.64	58.29	**86.22**	85.04	72.49	59.60	59.76	85.97	-	66.83	-

TABLE VIII
EXPERIMENTAL RESULTS ON HOUSE C WITH NULL CLASS

Feature	NB	HMM	HSMM	1NN	SVM	MLP	C4.5	SAE	DBN	CNN1d	CNN	LSTM1d	LSTM
Binary	40.70	24.84	29.55	25.78	40.44	36.97	33.34	45.16	**47.74**	45.10	42.36	43.71	40.24
Numeral	41.44	27.37	32.31	30.82	42.70	45.29	37.68	44.57	47.07	**48.68**	-	47.38	-
Probability	10.54	10.12	11.09	25.99	16.46	15.48	24.97	43.81	44.51	**46.04**	-	43.81	-

For example, CNN1d obtains the accuracy of 94.51%, 94.83%, and 92.98% compared to the 94.14%, 62.03%, and 61.44% of DBN and the 90.43%, 88.28%, and 60.46% of SAE. In addition, we observe that CNN1d achieves the best results and generally outperforms CNN, LSTM1d and LSTM. This indicates the existence of non-linear relationships among original features and the power of CNN in learning complex features. This is probably because the 2D feature encoding leads to sparse input that makes it difficult to extract meaningful information and the temporal relations that can be used by LSTM1d is limited. Furthermore, except DBN, we observe that using the numeral representation generally obtains better performance or comparable results to the binary and probability representations in the majority of cases. For example, NB obtains accuracy of 86.53% for the numeral representation, comparable to the 86.53% accuracy of binary representation and the 83.36% accuracy of probability representation. LSTM1d achieves accuracy of 93.90% for the numeral representation, which is slightly better than other two. The possible reason is that the binary representation suffers from loss of information and the probability representation has difficulty in parameter updating with different batches of the training data. Thus, compared with the numeral representation, the binary representation has difficulty in recognizing activities that trigger the same group of sensors and the probability representation leads to suboptimal solutions. Similar results can be observed from Tables IV and V.

E. Experimental Results With NULL Class

It seems that we obtained satisfactory activity recognition performance according to the above results, but this presents the ideal case where one only performs the predefined activities of interest. Unfortunately, it is inapplicable to the real-world applications, since the NULL class is inherently accompanied with human behavior and it is not trivial to filter out the NULL class. For example, 12.69% samples of the first smart home are associated with NULL class, and the numbers are 7.96% and 19.97% on the second and third smart homes, respectively. Herein, we consider the NULL class and comparatively evaluate it. Tables VI-VIII show the corresponding results. From the results, we observe that original feature encodings influence the performance of

an activity recognizer. For example, on the second dataset, when using NB, the three representations obtain the accuracy of 80.88%, 80.50% and 64.22%, respectively. On the third dataset, LSTM1d obtains the accuracy of 43.71%, 47.38%, and 43.81% with the three feature representations, respectively. We also observe CNN1d tends to obtain better results and outperforms CNN, LSTM1d, and LSTM. Besides, we observe that a majority of activity recognizers except DBN benefit from the numeral representation. For example, on the third smart home, when we use the numeral representation, CNN1d gets accuracy of 48.68%, which is higher than the other two cases (45.10% and 46.04%), and LSTM1d obtains the best accuracy of 47.38%. This indicates the discriminant ability of the use of numeral representation.

Furthermore, when comparing the results of without NULL class (Tables III-V) and with NULL class (Tables VI-VIII), we observe that the inclusion of NULL class generally lowers the recognition accuracy of an activity recognizer. For example, on the first dataset, NB obtains the accuracy of 86.53%, 86.34%, and 83.36% for the three representations, respectively, but they reduce to 76.90%, 77.03%, and 73.03% when the NULL class is considered. For CNN1d, the accuracy reduces to 85.30%, 85.79%, and 84.88% from 94.51%, 94.83%, and 93.60%, respectively. To present a better comparison, Figs. 6-8 show the results of the three different feature encodings on the datasets. We observe that the inclusion of NULL class deteriorates the performance of both deep learning and shallow models. It is mainly because it becomes difficult to recognize the increased number of classes and the newly added class probably share similar characteristics to existing classes such as the transition between two activities. Similar results can be obtained for the case of binary and probability representations.

Overall, according to the above results, we conclude that the choice of feature encodings largely determines the performance of a model and that the numerical representation generally leads to better performance. Although deep learning models can learn features from the raw signals, they have different discriminant capabilities and are applicable to different problems. Due to the sparsity of the multichannel sensor events, using a 1D vector to encode sensor data remains a priority. In addition, results show that the inclusion of NULL class lowers the recognition accuracy, which indicates that we should stress its existence in designing practical activity recognition supported applications.

V. Conclusion

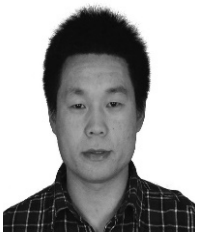
Automatically recognizing activities of daily living bridges the gap between the low-level sensor data and high-level applications and plays a key role in human-centric applications, especially the ambient assisted living systems. Environment sensor-based methods have advantages of non-intrusiveness and low cost and they provide a promising way for unobtrusive activity recognition. Since human activities are characterized by inherent complexity, how to accurately recognize ADLs is meaningful yet challenging. Although deep learning models have the end-to-end capability to learn features and provide enhanced performance in numerous fields, there are several issues that need further investigation when applying them to build in-home activity recognizers with environment sensors. In this study, we explore how to encode the multichannel streaming binary sensor events and evaluate their combinations with deep learning models, where we propose two different methods to process sensor data. Comparative experiments are conducted on public smart home datasets collected with binary sensors. Besides, we include other seven shallow classification models as a comparison. Experimental results indicate that different feature encodings influence both deep learning and shallow models and show the superiority of deep learning. Furthermore, we experimentally evaluate the influence of NULL class on an activity recognizer, which shows that its inclusion deteriorates the performance of deep learning and shallow models. This motivates researchers, to a certain extent, to consider the NULL class in developing and evaluating activity recognition enabled systems with environment sensors and even wearable sensors or cameras.

Although this study performs offline evaluation, both online and offline activity recognition involves the optimization of an activity recognizer. This study provides an objective metric for evaluating activity recognizers as most previous studies have done. Also, there are many applications such as long-term behavior pattern mining and daily health evaluation that have tolerance to time-delay. For the future work, we plan to conduct researches along with the following lines. First, as for the applicability of deep learning models to smart home scenarios, we can provide the activity recognition service via the locally deployed high-performance servers or a public cloud platform. Particularly, along with the development of edge computing, we could implement the activity recognition engine inside edge devices to better support edge intelligence. Second, due to the impact of NULL class, how to discover and recognize new activities from the NULL class in a specific application is of much value in improving an activity recognizer, which remains another research topic. Particularly, we can combine activity discovery with activity recognition, where the former detects emerging activities to help the latter better adapt to a dynamic environment.

References

[1] S. D. T. Kelly, N. K. Suryadevara, and S. C. Mukhopadhyay, "Towards the implementation of IoT for environmental condition monitoring in homes," *IEEE Sensors J.*, vol. 13, no. 10, pp. 3846–3853, Oct. 2013.
[2] D. J. Cook, A. S. Crandall, B. L. Thomas, and N. C. Krishnan, "CASAS: A smart home in a box," *Computer*, vol. 46, no. 7, pp. 62–69, Jul. 2013.
[3] A. Wang, G. Chen, J. Yang, S. Zhao, and C.-Y. Chang, "A comparative study on human activity recognition using inertial sensors in a smartphone," *IEEE Sensors J.*, vol. 16, no. 11, pp. 4566–4578, Jun. 2016.
[4] A. Haque, A. Milstein, and L. Fei-Fei, "Illuminating the dark spaces of healthcare with ambient intelligence," *Nature*, vol. 585, no. 7824, pp. 193–202, Sep. 2020.
[5] N. K. Suryadevara and S. C. Mukhopadhyay, "Determining wellness through an ambient assisted living environment," *IEEE Intell. Syst.*, vol. 29, no. 3, pp. 30–37, May 2014.
[6] P. N. Dawadi, D. J. Cook, and M. Schmitter-Edgecombe, "Automated cognitive health assessment from smart home-based behavior data," *IEEE J. Biomed. Health Informat.*, vol. 20, no. 4, pp. 1188–1194, Jul. 2016.
[7] G. Chen, A. Wang, S. Zhao, L. Liu, and C.-Y. Chang, "Latent feature learning for activity recognition using simple sensors in smart homes," *Multimedia Tools Appl.*, vol. 77, no. 12, pp. 15201–15219, Jun. 2018.
[8] N. C. Krishnan and D. J. Cook, "Activity recognition on streaming sensor data," *Pervas. Mobile Comput.*, vol. 10, pp. 138–154, Feb. 2014.

[9] F. Hussain, F. Hussain, M. Ehatisham-ul-Haq, and M. A. Azam, "Activity-aware fall detection and recognition based on wearable sensors," *IEEE Sensors J.*, vol. 19, no. 12, pp. 4528–4536, Jun. 2019.
[10] A. Wang, G. Chen, X. Wu, L. Liu, N. An, and C.-Y. Chang, "Towards human activity recognition: A hierarchical feature selection framework," *Sensors*, vol. 18, no. 11, p. 3629, Oct. 2018.
[11] A. Bulling, U. Blanke, and B. Schiele, "A tutorial on human activity recognition using body-worn inertial sensors," *ACM Comput. Surv.*, vol. 46, no. 3, p. 33, 2014.
[12] C. Dhiman and D. K. Vishwakarma, "A robust framework for abnormal human action recognition using $\mathcal{R}$-transform and zernike moments in depth videos," *IEEE Sensors J.*, vol. 19, no. 13, pp. 5195–5203, Jul. 2019.
[13] T. van Kasteren, "Activity recognition for health monitoring elderly using temporal probabilistic models," Ph.D. dissertation, Inform. Inst., Univ. Amsterdam, Amsterdam, The Netherlands, 2011.
[14] X. Qingxin, A. Wada, J. Korpela, T. Maekawa, and Y. Namioka, "Unsupervised factory activity recognition with wearable sensors using process instruction information," *Proc. ACM Interact., Mobile, Wearable Ubiquitous Technol.*, vol. 3, no. 2, pp. 1–23, Jun. 2019.
[15] H. K. Jang, H. Han, and S. W. Yoon, "Comprehensive monitoring of bad head and shoulder postures by wearable magnetic sensors and deep learning," *IEEE Sensors J.*, vol. 20, no. 22, pp. 13768–13775, Nov. 2020.
[16] H. Y. Yatbaz, S. Eraslan, Y. Yesilada, and E. Ever, "Activity recognition using binary sensors for elderly people living alone: Scanpath trend analysis approach," *IEEE Sensors J.*, vol. 19, no. 17, pp. 7575–7582, Sep. 2019.
[17] A. Shrestha, H. Li, J. Le Kernec, and F. Fioranelli, "Continuous human activity classification from FMCW radar with bi-LSTM networks," *IEEE Sensors J.*, vol. 20, no. 22, pp. 13607–13619, Nov. 2020.
[18] G. E. Hinton, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.
[19] F. Ordóñez and D. Roggen, "Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition," *Sensors*, vol. 16, no. 1, p. 115, Jan. 2016.
[20] G. Acampora, D. J. Cook, P. Rashidi, and A. V. Vasilakos, "A survey on ambient intelligence in healthcare," *Proc. IEEE*, vol. 101, no. 12, pp. 2470–2494, Dec. 2013.
[21] M. Gochoo, T.-H. Tan, S.-H. Liu, F.-R. Jean, F. S. Alnajjar, and S.-C. Huang, "Unobtrusive activity recognition of elderly people living alone using anonymous binary sensors and DCNN," *IEEE J. Biomed. Health Informat.*, vol. 23, no. 2, pp. 693–702, Mar. 2019.
[22] T. Plötz, N. Y. Hammerla, and P. L. Olivier, "Feature learning for activity recognition in ubiquitous computing," in *Proc. 22nd Int. Joint Conf. Artif. Intell.*, 2011, pp. 1729–1734.
[23] Y. Liu, L. Nie, L. Liu, and D. S. Rosenblum, "From action to activity: Sensor-based activity recognition," *Neurocomputing*, vol. 181, pp. 108–115, Mar. 2016.
[24] G. Okeyo, L. Chen, H. Wang, and R. Sterritt, "Dynamic sensor data segmentation for real-time knowledge-driven activity recognition," *Pervas. Mobile Comput.*, vol. 10, pp. 155–172, Feb. 2014.
[25] L. Chen, C. D. Nugent, and H. Wang, "A knowledge-driven approach to activity recognition in smart homes," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 6, pp. 961–974, Jun. 2012.
[26] F. J. Ordóñez, P. de Toledo, and A. Sanchis, "Activity recognition using hybrid Generative/Discriminative models on home environments using binary sensors," *Sensors*, vol. 13, no. 5, pp. 5460–5477, Apr. 2013.
[27] M. Z. Uddin and M. M. Hassan, "Activity recognition for cognitive assistance using body sensors data and deep convolutional neural network," *IEEE Sensors J.*, vol. 19, no. 19, pp. 8413–8419, Oct. 2019.
[28] Q. Teng, K. Wang, L. Zhang, and J. He, "The layer-wise training convolutional neural networks using local loss for sensor-based human activity recognition," *IEEE Sensors J.*, vol. 20, no. 13, pp. 7265–7274, Jul. 2020.
[29] C. A. Ronao and S.-B. Cho, "Human activity recognition with smartphone sensors using deep learning neural networks," *Expert Syst. Appl.*, vol. 59, pp. 235–244, Oct. 2016.
[30] F. Ordóñez and D. Roggen, "Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition," *Sensors*, vol. 16, no. 1, p. 115, Jan. 2016.
[31] Y. Guan and T. Ploetz, "Ensembles of deep LSTM learners for activity recognition using wearables," *ACM Interact., Mobile, Wearable Ubiquitous Technol.*, vol. 1, no. 2, p. 11, Mar. 2017.
[32] D. Liciotti, M. Bernardini, L. Romeo, and E. Frontoni, "A sequential deep learning application for recognising human activities in smart homes," *Neurocomputing*, vol. 396, pp. 501–513, Jul. 2020.



Aiguo Wang received the B.S. and Ph.D. degrees from the Hefei University of Technology, China, in 2010 and 2015, respectively.

He is currently a Distinguished Researcher with the School of Electronic Information Engineering, Foshan University, China. His current research interests include machine learning, data mining, and pervasive computing.

Shenghui Zhao received the M.S. degree from the Hefei University of Technology, China, in 2003, and the Ph.D. degree from Southeast University, China, in 2013.

She is currently a Professor with the School of Computer and Information Engineering, Chuzhou University, China. Her current research interests include trusted computing, healthcare, and the Internet of Things.

Chundi Zheng received the M.S. degree from Xidian University, Xi'an, China, in 2006, and the Ph.D. degree from Tsinghua University, Beijing, China, in 2013.

He is currently an Associate Professor with the School of Electronic Information Engineering, Foshan University, China. His research interests include signal processing and sparse recovery.

Jing Yang received the B.S. and Ph.D. degrees from the Hefei University of Technology, China, in 2004 and 2013, respectively.

She is currently an Associate Professor with the School of Computer and Information Engineering, Hefei University of Technology, China. Her current research interests include artificial intelligence, data mining, and Bayesian networks.

Guilin Chen received the B.S. degree from Anhui Normal University, China, in 1985, and the M.S. degree from the Hefei University of Technology, in 2007.

He is currently a Professor with the School of Computer and Information Engineering, Chuzhou University, China. His current research interests include cloud computing, wireless networks, healthcare, and the Internet of Things.

Chih-Yung Chang (Member, IEEE) received the Ph.D. degree in computer science and information engineering from National Central University, Zhongli, Taiwan, in 1995.

He is currently a Full Professor with the Department of Computer Science and Information Engineering, Tamkang University, Taipei, Taiwan. His current research interests include the Internet of Things, wireless sensor networks, ad hoc wireless networks, and long-term evolution broadband technologies. He served as an Associate Guest Editor for several SCI-indexed journals, including the *International Journal of Ad Hoc and Ubiquitous Computing* from 2011 to 2014, the *International Journal of Distributed Sensor Networks* from 2012 to 2014, *IET Communications* in 2011, *Telecommunication Systems* in 2010, the *Journal of Information Science and Engineering* in 2008, and the *Journal of Internet Technology* from 2004 to 2008.