CapMatch: Semi-supervised Contrastive Transformer Capsule with Feature-based Knowledge Distillation for Human Activity Recognition

Zhiwen Xiao, *Member, IEEE*, Huagang Tong, Rong Qu, *Senior Member, IEEE*, Huanlai Xing, *Member, IEEE*, Shouxi Luo, *Member, IEEE*, Zonghai Zhu, Fuhong Song, and Li Feng

***Abstract*—This paper proposes a semi-supervised contrastive capsule transformer method with feature-based knowledge distillation (KD) that simplifies existing semisupervised learning (SSL) techniques for wearable human activity recognition (HAR), called CapMatch. CapMatch gracefully hybridizes supervised learning and unsupervised learning to extract rich representations from input data. In unsupervised learning, CapMatch leverages the pseudo-labeling, contrastive learning (CL), and feature-based KD techniques to construct similarity learning on lower- and higher-level semantic information extracted from two augmentation versions of the data, "weak" and "timecut", to recognize the relationships among the obtained features of classes in the unlabeled data. CapMatch combines the outputs of the weak- and timecut-augmented models to form pseudo-labeling and thus CL. Meanwhile, CapMatch uses the feature-based KD to transfer knowledge from the intermediate layers of the weak augmented model to those of the timecut augmented model. To effectively capture both local and global patterns of HAR data, we design a capsule transformer network consisting of four capsule-based transformer blocks and one routing layer. Experimental results show that compared with a number of state-of-the-art semi-supervised and supervised algorithms, the proposed CapMatch achieves decent performance on three commonly used HAR datasets, namely, HAPT, WISDM, and UCI_HAR. With only 10% of data labeled, CapMatch achieves F_1 values of higher than 85.00% on these datasets, outperforming 14 semi-supervised algorithms. When the proportion of labeled data reaches 30%, CapMatch obtains F_1 values of no lower than 88.00% on the datasets above, which is better than several classical supervised algorithms, e.g., decision tree and KNN.**

***Index Terms*—Capsule Network, Contrastive Learning, Human Activity Recognition, Knowledge Distillation, Semi-supervised Learning, Similarity Learning, Wearable Sensors**

Manuscript received 7, Oct., 2022; revised 8, Sep., 2023 and 11, Qct., 2023; accepted 10, Dec., 2023. This work was partially supported by the Natural Science Foundation of Hebei Province (No. F2022105027), the Humanities and Social Sciences Program of the Ministry of Education (No. 23YJCZH201), the Natural Science Foundation of Sichuan Province (No. 2022NSFSC0568, No. 2022NSFSC0944, and No. 2023NSFSC0459), and the Fundamental Research Funds for the Central Universities, P. R. China (Corresponding Author: Huanlai Xing).

Z. Xiao, H. Xing, S. Luo, Z. Zhu, and L. Feng are with the School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu 610031, China, also with the Tangshan Institute of Southwest Jiaotong University, Tangshan 063000, China, and also with the Engineering Research Center of Sustainable Urban Intelligent Transportation, Ministry of Education, China (Emails: xiao1994zw@163.com; hxx@home.swjtu.edu.cn; sxluo@swjtu.edu.cn; zzhu@swjtu.edu.cn; fengli@swjtu.edu.cn).

H. Tong is with College of Economic and Management, Nanjing Tech University, Puzhu Road(S), Nanjing 211816, China (Email: huagangtong@gmail.com).

R. Qu is with the School of Computer Science, University of Nottingham, Nottingham NG7 2RD 455356, UK (Email: rong.qu@nottingham.ac.uk).

F. Song is with the School of Information, Guizhou University of Finance and Economics, Guiyang 550025, China (Email: fhsong@mail.gufe.edu.cn).

I. INTRODUCTION

HUMAN activity recognition (HAR) identifies people's actions based on their observations and environmental surroundings [1]. HAR has been widely used in various real-world domains, such as electroencephalography (EEG) analysis [2], gesture detection [3], and healthcare system [4]. With the prevalence of mobile devices, e.g., smartphones and watches, wearable HAR data collection has become accessible and convenient. Thus, wearable sensor-based HAR has grown into one of the mainstream research topics in HAR [5]. Wearable HAR data is a series of time-ordered data points collected by wearable sensor(s), e.g., triaxial accelerometer owns thee sensors producing X-, Y-, and Z-axis data simultaneously. Such a series is associated with a single or multiple time-dependent variables, i.e., univariate and multivariate [6]. A HAR algorithm captures the local and global patterns from a given time series, e.g., those associated with one variable and those across multiple variables [7], [8].

Over the years, a large number of algorithms have been developed to address wearable sensor-based HAR problems, mainly through traditional and deep learning techniques [5], [6], [7]. Traditional algorithms are usually statistical or machine learning method based, which focus on capturing shallow features from HAR data. For example, Zhu and Sheng [9] introduced a hierarchical hidden Markov model for context-based recognition. Chen *et al.* [10] proposed a HAR system with coordinate transformation and principal component analysis (PCA) and online support vector machine (SVM). In contrast, deep learning ones are able to extract the intrinsic connections among representations by constructing the internal representation hierarchy of data [11], e.g., Al-qaness *et al.* [12] designed a multilevel residual network with attention for HAR feature extraction. Xia *et al.* [13] put forward a multiple-level domain adaptive learning model that used a single inertial measurement unit sensor to obtain accurate activity recognition. Shu *et al.* [14] presented a graph long short-term memory (LSTM)-in-LSTM method for group activity recognition, where person-level actions and group-level activity were modeled simultaneously. Unfortunately, all the algorithms above heavily relied on labeled data that usually consumed an incredible amount of human resource cost for raw data annotation.

Semi-supervised learning (SSL) leverages a small amount of labeled data to capture features from a dataset with a large amount of unlabeled data [15]. To mitigate the dependency on labeled data, SSL-based HAR has attracted increasingly more research interests. The SSL algorithms for HAR can be roughly classified into three categories: graph-based, self-labeled, and self-supervised. Graph-based algorithms use graph techniques to learn the similarity between the feature maps obtained from the HAR data, e.g., the multi-graph-based SSL [16], shared structure discovery SSL [17], and dynamic graph-based SSL [18]. Self-labeled ones usually adopt a supervised classifier to label instances with the unknown class without any specific input data suppositions, mainly relying on two methods: co-training [19], [20] and self-training [21], [22], [23]. Self-supervised algorithms consider a model's class prediction as a pseudo label of the training object, e.g., SelfHAR [24], CSSHAR [25], and ColloSSL [26]. In summary, the algorithms above usually use one or two of the four main SSL techniques, namely, entropy minimization, consistency regularization, pseudo-labeling, and generic regularization. Unfortunately, most SSL algorithms for HAR ignore integration of these techniques and thus have limited ability to capture rich representations from the data.

Recently, algorithms hybridizing multiple SSL techniques have become prevalent in the semi-supervised image classification community. For example, MixMatch used a single loss to integrate the entropy minimization, consistency regularization, and generic regularization [27]. ReMixMatch was an improved version of MixMatch, with distribution alignment and augmentation anchoring as two additional techniques [28]. FixMatch took the advantages of ReMixMatch and pseudo-labeling to capture sufficient representations from input data [29]. Zhang *et al.* [30] integrated curriculum pseudo-labeling into FixMatch to form FlexMatch for semi-supervised image classification.

The ensemble algorithms above, e.g., FixMatch and its variants (e.g., FlexMatch), usually consist of supervised learning based on a small amount of labeled data and unsupervised learning based on a large amount of unlabeled data. Their performance heavily depends on the representation learning on the unlabeled data, where lower- and higher-level semantic information is of significant importance [31]. However, most of the ensemble algorithms only emphasize the similarity learning on higher-level semantic information, ignoring the importance of lower-level semantic information on representation learning, which limits their ability of extracting abundant representations from unlabeled data. For example, the similarity learning in FixMatch only combines the output extracted from "weak" data and that from "strong" data through pseudo-labeling, where "weak" and "strong" are two augmentation versions of the same data. *Indeed, the performance of an algorithm is heavily dependent on the quality of lower- and higher-level semantic information obtained from the data through instance-level representation learning [31]. Therefore, it is crucial for an SSL algorithm to enhance its similarity learning on both lower- and higher-level semantic information, which ensures the algorithm's performance in unsupervised learning.*

Recently, feature-based knowledge distillation (KD), an effective form of similarity learning on lower-level semantic information, has emerged. This technique enables knowledge flow between the intermediate layers of a teacher and those of a student, helping the student obtain decent performance on instance-level representation learning [32]. On the other hand, contrastive learning (CL), a popular self-supervised learning method, studies the similarity between different views from the same sample and the similarity between the views from different samples, which improves the quality of the learned representations and thus provides rich semantic information for downstream tasks [33].

On the other hand, most SSL algorithms for HAR, e.g., ActSemiCNN [22], CSSHAR [25], and ColloSSL [26], usually use neural networks to capture features from the input. Neural networks, however, easily cause potential information loss of entities/objects due to the intrinsic translation invariance, e.g., Maxpooling. To overcome the drawback above, Sabour *et al.* [34] introduced a capsule network (CapNet) with dynamic routing mechanism to obtain entities' semantic information, e.g., location and orientation. It was reported that CapNet was quite effective in mining sufficient lower- and higher-level semantic information.

Based on FixMatch, we introduce the feature-based KD, CL and capsule-based methods to design a semi-supervised contrastive transformer capsule model for wearable HAR, called CapMatch. This model gracefully integrates supervised and unsupervised learning to mine rich representations from partially labeled data. Like most supervised capsule algorithms [35], [36], [37], CapMatch guides the prediction vectors towards the corresponding ground labels on the labeled data. On the other hand, CapMatch leverages data augmentation, pseudo-labeling, CL, and feature-based KD techniques to recognize the relationships among the features of classes obtained from the unlabeled data. CapMatch generates different views of the same sample by two data augmentation methods, namely "weak" and "timecut". Similarity learning on the lower- and higher-level semantic information extracted from the two types of augmented data is established in unsupervised learning. Meanwhile, CapMatch uses feature-based KD to transfer knowledge from the intermediate layers of weak-augmented model to those of the timecut-augmented model. The overview of CapMatch is shown in Figure 1.

Our significant contributions are summarized below.

- We propose a capsule transformer network with four capsule-based transformer blocks and one routing layer as the CapMatch's feature extractor in Figure 1. Unlike the vanilla transformer [38], the capsule-based transformer block considers the interaction rules among capsules, helping CapMatch mine abundant valuable connections and regularizations from the HAR data, e.g., the length of each capsule's vector is the capsule's entities' probability.
- CapMatch applies the pseudo-labeling, CL, and feature-based KD techniques to constructing similarity learning on the lower- and higher-level semantic information extracted from the weak and timecut versions of input data, resulting in high-quality feature extraction performance.

- Experimental results show that CapMatch outperforms 14 SSL algorithms on three widely used HAR datasets: the smartphone-based recognition of human activities and postural transitions dataset (HAPT), wireless sensor data mining (WISDM), and University of California Irvine (UCI) HAR using smartphones (UCI_HAR) when the labeled data takes up 10%, 20%, and 30% of the training data, respectively. In particular, CapMatch overweighs a few supervised algorithms on these datasets in terms of F_1 value, e.g., decision tree and k-nearest neighbor (KNN), when the labeled data account for 30% of the training data.

The rest of the paper is organized as follows. Section II reviews the related work. Section III describes the CapMatch's overall structure and its components. Section IV analyzes the experimental results, and Section V draws the conclusion.

II. Related Work

This section reviews the existing studies on wearable HAR, capsule network, CL, and KD.

A. Wearable HAR Algorithms

There have been many algorithms for addressing various wearable sensor-based HAR problems. These algorithms are either traditional or deep learning based [5], [6], [7]. Traditional algorithms usually use statistical or machine learning methods to mine shallow features from HAR data, such as PCA, SVM, KNN, Bagging, logistic regression (LR), Bayes algorithm, J48, Markov regression, collaboration algorithm, logic-based reasoning, and Fuzzy algorithm [9], [10], [39], [40], [41], [42].

On the other hand, deep learning algorithms can extract not only the shallow features but also the intrinsic regularizations and connections hidden in the data [11]. For example, Ravi *et al.* [43] introduced a temporal convolutional model for activity recognition on low-power smartphones. Zhang *et al.* [44] proposed a multi-head convolutional attention network to capture multi-scale features from HAR data. Besides, the stacked denoising autoencoder [45], graph-based LSTM-in-LSTM [14], multilevel residual network with attention [12], kernel density estimation-based model [46], Lego CNN [47], deformable convolutional network [48], multiple-level domain adaptive learning model [49], selective kernel convolution [50], and CNN-LSTM-based model [51] are all well-known HAR algorithms based on deep neural networks.

B. Capsule Network

Capsule network was developed to solve the problem of information loss of entities/objects due to translation invariance, e.g., Maxpooling [34]. In just a few years, capsule-based models have attracted increasingly more research efforts. For example, Chen *et al.* [35] proposed a contemporary novel neural network capsule architecture with multi-dimension and abundant spatial information for fault diagnosis. Feng *et al.* [36] presented a dual-routing capsule graph neural network to capture temporal and spatial features from video data. Xiao *et al.* [37] devised a multi-process collaborative capsule architecture for multi-scale feature extraction on time series classification. Saad and Chen [52] designed an efficient capsule network for Seismic Phase prediction. Sun *et al.* [53] put forward a capsule and gate recurrent unit network to recognize human activities.

C. Contrastive Learning

As one of the most effective representation learning techniques, CL is committed to exploring the differences between different views from the same sample and the significant differences among different samples, providing sufficient representations for downstream tasks [33]. He *et al.* [54] introduced an unsupervised visual learning algorithm with a momentum encoder, called MoCo, to explore the relationships among different samples. Based on MoCo, Chen *et al.* [55] designed a simple method called SimCLR to learn the representations from antagonistic pairs. Han *et al.* [56] proposed an unsupervised structure-adaptive graph CL method that explored saliency regularizations and relationships from the input data. On the other hand, CL has been widely applied to tackle various real-world problems. For instance, Feng *et al.* [57] adopted a CL-based monocular object detection model to distinguish 3-dimensional objects. In [58], an intra- and inter-Slice CL network was used to address OCT fluid segmentation problems. In [59], a CL-based joint learning framework was applied to accurate COVID-19 identification. In [60], a contrastive SSL method was utilized to capture the representations from remote sensing data. With the help of CL, pre-trained language models accelerated their fine-tuning phase and improved their generalization abilities [61]. Liu *et al.* [62] devised a contrastive self-supervise learning method for anomaly detection. Yu *et al.* [63] designed a weakly supervised CL framework with domain adaptation for vehicle reidentification.

D. Knowledge Distillation

KD, one of the most popular regularization techniques, encourages knowledge transfer from a cumbersome network (i.e., teacher) to a lightweight one (i.e., student). According to the knowledge form, researchers roughly divide the existing KD algorithms into three categories: response-based, feature-based, and relation-based [32]. The response-based method transfers the knowledge from the output (i.e., logits) of a teacher to that of a student [64]. For example, Feng *et al.* [65] introduced a resolution-aware KD method to transfer high-level semantic information to low-level one. In [66], an expert embedding KD method was used to enhance the knowledge capacity during the knowledge transfer process. In [67], a collaborative KD algorithm was utilized to improve the accuracy of image classification.

The feature-based method enables knowledge sharing between intermediate layers of a teacher and its student instead of output-to-output. Since the pioneering work FitNet [68], a considerable amount of feature-based studies have been conducted to solve various application tasks. For instance, Zhang *et al.* [69] designed an effective KD method based on

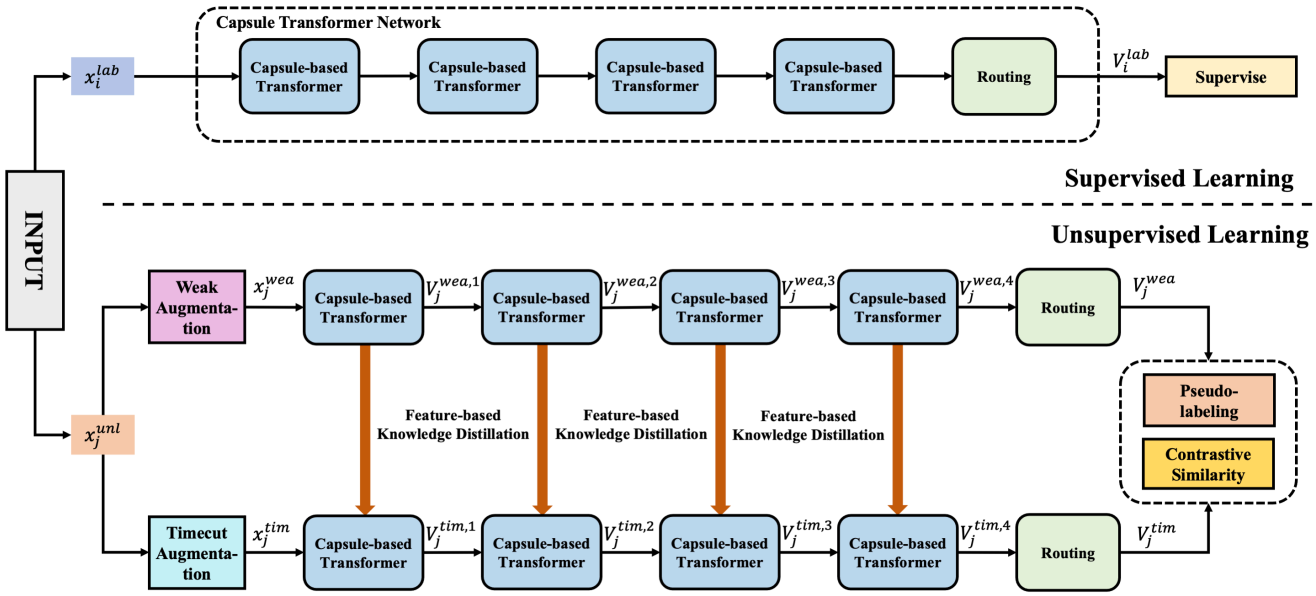


Fig. 1. The overview of CapMatch. The capsule transformer network consists of four capsule-based transformer blocks and a routing layer.

layer-calibration and task-disentangle distillation for remote sensing. Hao *et al.* [70] presented a collaborative data-free framework with multi-level feature sharing for multi-scale feature extraction. Different from response- and feature-based methods, the relation-based method pays more attention to capturing the relationships among layers in the student and teacher models. Yu *et al.* [71] devised a relation-based metric learning model to improve the quality of image feature embedding. Guo *et al.* [72] introduced a multi-level attention-based KD approach to capture potential correlations between the teacher and student models. In [73], a cross-layer mutual distillation method enabled sufficient knowledge from the teacher to the student, improving the feature extraction ability of the student model.

III. CapMatch

This section first overviews the CapMatch's structure. Then it describes the problem formulation, capsule-based transformer, routing, data augmentation, CL, feature-based KD, and loss function one by one.

A. Overview

CapMatch contains supervised and unsupervised learning processes, as shown in Figure 1. The capsule transformer network, composed of four capsule-based transformer blocks and one routing layer, is the model's feature extractor. CapMatch guides the prediction vectors towards the corresponding ground labels by a margin loss function on the labeled data in the supervised learning process. On the other hand, CapMatch leverages data augmentation, pseudo-labeling, CL, and feature-based KD techniques to recognize the relationships among the features of classes obtained from the unlabeled data in the unsupervised learning process. Two data augmentation methods, namely "weak" and "timecut", are used to generate different views of the same sample. CapMatch establishes similarity learning on the lower- and higher-level semantic information extracted from the weak and timecut versions of the data to enhance the instance-level representation learning in unsupervised learning. To be specific, CapMatch not only allows the weak-augmented artificial labels to supervise the timecut-augmented prediction vectors by a margin loss function but also combines the outputs of the weak- and timecut-augmented models to form similarity learning by a CL loss function. Meanwhile, CapMatch promotes the knowledge flow from the intermediate layers of weak-augmented model to those of the timecut-augmented model via feature-based KD.

B. Problem Formulation

Assume $x_i = \{\{x_{1,1}^{(i)}, ..., x_{1,d}^{(i)}\}, \, ..., \{\{x_{l,1}^{(i)}, ..., x_{l,d}^{(i)}\}\} \in \mathcal{X}$ is an arbitrary HAR time-series, where $\mathcal{X} \subseteq \mathbb{R}^{l \times d}$ is the input space, and l and d denote the length and dimension of x_i, respectively. $y_i \in \mathcal{Y}$ is a categorical variable associated with x_i, where $\mathcal{Y}$ is the target space. We aim at training a prediction model $\mathcal{M} : \mathcal{X} \mapsto \mathcal{Y}$ on an arbitrary dataset, $\mathcal{D} = \{\mathcal{D}_{train}, \mathcal{D}_{val}, \mathcal{D}_{test}\}$. $\mathcal{D}_{train} = \{\mathcal{D}_{train}^{lab}, \mathcal{D}_{train}^{unl}\}$, $\mathcal{D}_{val} = \{x_i, y_i\}_{i=1}^{n_{val}}$, and $\mathcal{D}_{test} = \{x_i, y_i\}_{i=1}^{n_{test}}$ are the data for training, validation, and testing, respectively, where

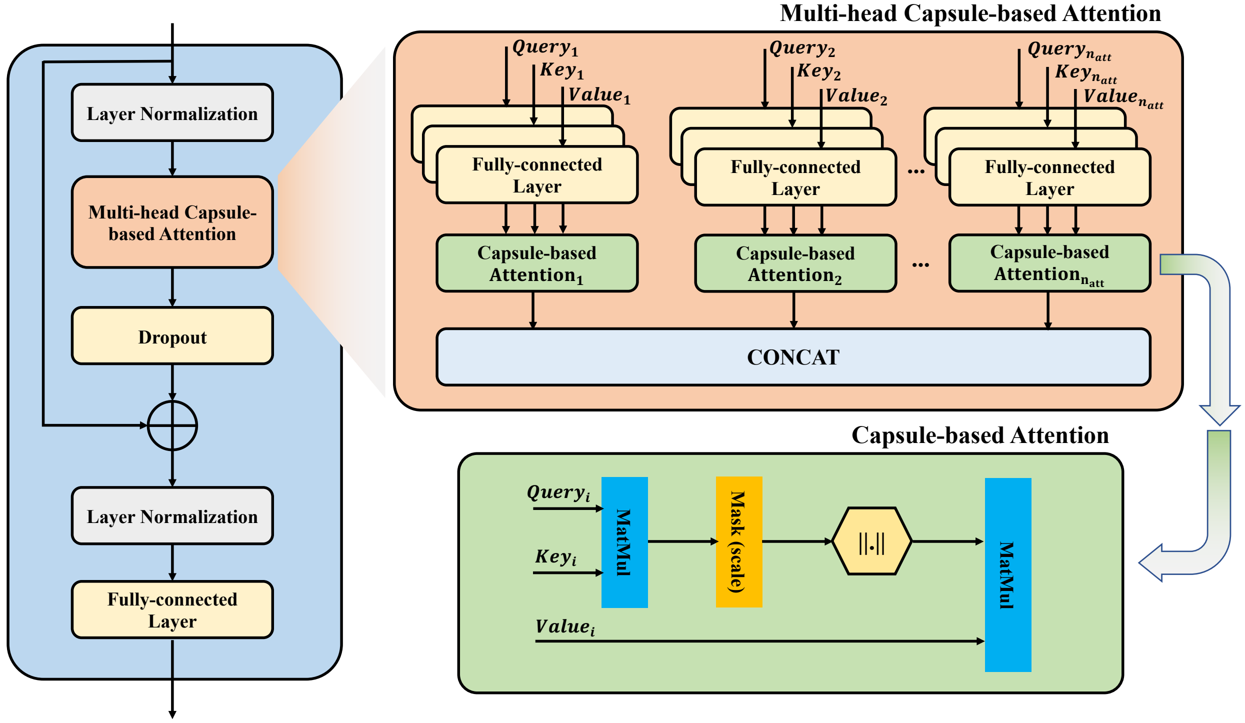


Fig. 2. Architecture of a capsule-based transformer block. Note: ||.|| outputs the length of a given vector and "MatMul" represents the matrix multiplication operation.

$\mathcal{D}^{lab}_{train} = \{x^{lab}_i, y_i\}^{n_{lab}}_{i=1}$ and $\mathcal{D}^{unl}_{train} = \{x^{unl}_j\}^{n_{unl}}_{j=1}$ are the labeled and unlabeled training data, respectively. n_{lab} and n_{unl} denote the sizes of labeled and unlabeled data, respectively. n_{val} and n_{test} are the sizes of validation and testing data, respectively.

C. Capsule-based Transformer

In the capsule transformer network, four capsule-based transformer blocks are adopted to capture local and global patterns of the HAR data, providing rich representations with routing.

Within the capsule community, there are also some transformer-based capsule studies. Most of these studies are primarily based on two methods, one uses the vanilla transformer [74], [75], and the other embeds the dynamic routing into the transformer to form an embedding transformer [76]. These transformer studies ignore the interaction rules among capsules, e.g., the length of each capsule's vector is the capsule's entities' probability, resulting in the absence of necessary information. To solve the issue above, this paper design a transformer-based capsule block following the interaction rules among capsules to relate the representations at different locations of the input data to extract the intrinsic connections and regularizations among the representations obtained, as shown in Figure 2. The multi-head capsule attention, containing n_{att} capsule-based attention modules, is the core of each transformer. Each capsule-based attention module, e.g., $Attention_i$, transfers a query, $Query_i$, and its key-value pairs, Key_i-$Value_i$, to an output, V^{att}_i. Different from the vanilla attention [38], the capsule-based attention considers

Algorithm 1 Routing

1: **procedure** ROUTING(s_j, n_{iter}) ▷ n_{iter} denotes the number of iterations.
2: Initialize weight matrix W_{ij};
3: Set $\hat{v}_{j|i} = W_{ij}s_j$ and $b_{ij} = 0$;
4: Routing
5: **for** n_{iter} iterations **do**
6: Obtain k_{ij} using Eq. (4);
7: Obtain $\hat{v}_{j|i}$ and s_j using Eq. (3);
8: Obtain b_{ij} using Eq. (5);
9: **end for**
10: **return** v_j;
11: **end procedure**

the interaction rules among capsules, e.g., the length of each capsule's vector is the capsule's entities' probability. V^{att}_i is defined as:

$$V^{att}_i = ||\frac{Query_i \cdot Key^T_i}{\sqrt{d_i}}|| \cdot Value_i \tag{1}$$

where, Key^T_i denotes the transpose of Key_i, d_i is the dimension of Key_i, and $||.||$ outputs the length of a given vector.

Let V_{mul} be the output of a multi-head capsule attention. V_{mul} fuses the n_{att} capsule-based attention through the CONCAT function, f_{concat}, to provide sufficient global features. V_{mul} is defined in Eq. (2).

$$V_{mul} = f_{concat}([V^{att}_1, V^{att}_2, ..., V^{att}_{n_{att}}]) \tag{2}$$

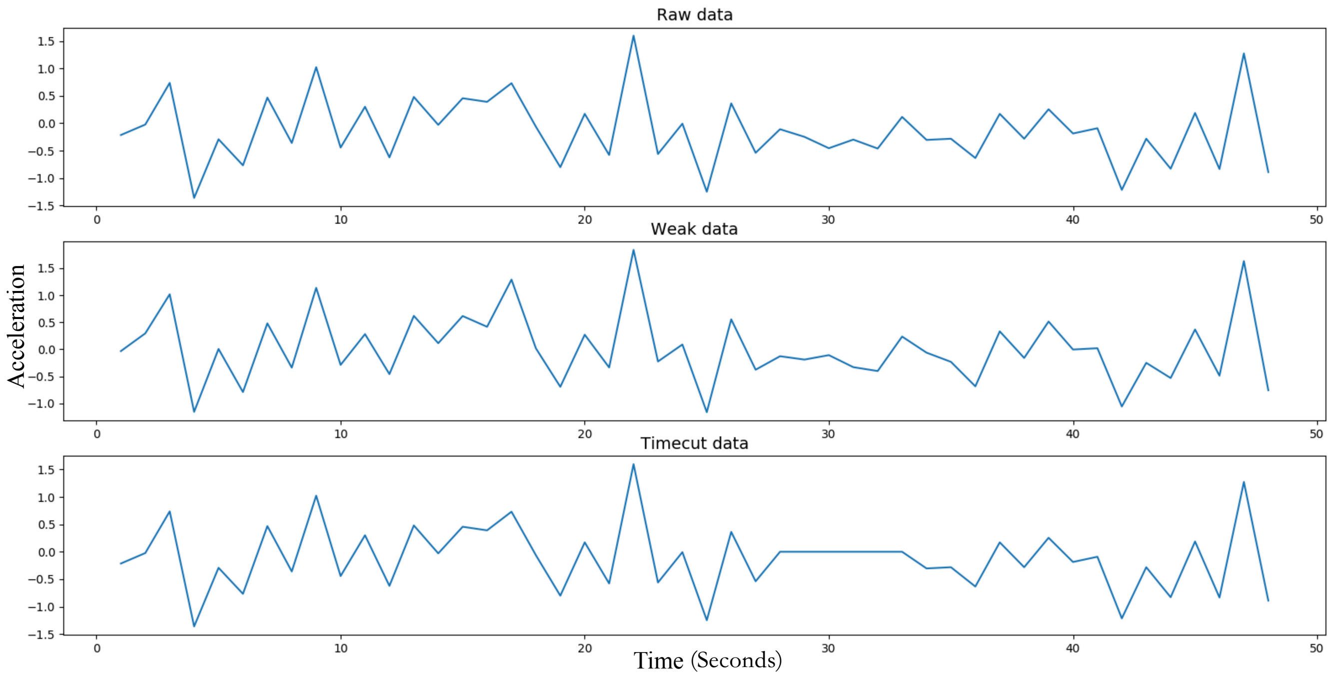


Fig. 3. Example raw data and its weak- and timecut-augmented data on the WISDM dataset.

D. Routing

Following [34], [35], [36], [37], we adopt the routing layer to promote the interaction among capsules, which helps mine the relationships among them. Given capsule j, its input s_j is defined in Eq. (3).

$$s_j = \sum_i k_{ij}\hat{v}_{j|i}, \quad \hat{v}_{j|i} = W_{ij}v_i \tag{3}$$

where, the prediction vector, $\hat{v}_{j|i}$, is obtained by multiplying the output of capsule i in the previous layer, v_i, by a weight matrix, W_{ij}. k_{ij} is a coupling coefficient between all capsules in the current layer and capsule i in the previous layer calculated by a softmax function, $f_{softmax}$, through an iterative routing process [34], [35], [37]. k_{ij} is calculated as:

$$k_{ij} = f_{softmax}(b_{ij}) \tag{4}$$

where, b_{ij} is the log prior probabilities that capsules i and j couple. b_{ij} measures the "agreement" between the current output v_j and the prediction $\hat{v}_{j|v}$, where $\hat{v}_{j|v}$ is obtained by capsule i from the previous layer. b_{ij} is defined in Eqs. (5)-(7).

$$b_{ij} = b_{ij} + v_j \cdot \hat{v}_{j|v} \tag{5}$$

$$v_j = f_{squash}(s_j) \tag{6}$$

$$f_{squash}(x) = \frac{||x||^2}{1+||x||^2}\frac{x}{||x||} \tag{7}$$

where, v_j is output of capsule j by "squashing" its input, s_j, in the current layer. The pseudo-code of Routing is shown in Algorithm 1.

E. Data Augmentation

Data augmentation is a widely used regularization method to efficiently improve a model's robustness in deep learning [27], [28], [29]. CapMatch leverages two augmentation methods, namely "weak" and "timecut", to produce different views from the same sample as the input in the unsupervised learning process. Specifically, the weak augmentation is realized via a jitter-and-scale strategy, e.g., adding the Gaussian function to the raw data. The timecut version is a modified version of Cutout [77] that transforms a small piece of the raw unlabeled data without changing its overall trend. Figure 3 shows an example of raw data and its weak- and timecut-augmented data on the WISDM dataset.

F. Feature-based Knowledge Distillation

Feature-based KD encourages knowledge transfer between intermediate layers of the teacher and student models, improving the student's feature extraction ability [32]. Let $V_j^{wea,1}$, $V_j^{wea,2}$, $V_j^{wea,3}$ and $V_j^{wea,4}$ denote the outputs of the four capsule-based transformer blocks associated with the "weak" augmented data, x_j^{wea}, respectively; let $V_j^{tim,1}$, $V_j^{tim,2}$, $V_j^{tim,3}$ and $V_j^{tim,4}$ be the outputs of the four capsule-based transformer blocks associated with the "timecut" augmented data, x_j^{tim}, respectively, in Figure 1.

The proposed KD loss, $\mathcal{L}_{KD}$, leverages an L_2 loss function to measure the differences between the features obtained from x_j^{wea} and those from x_j^{tim}. $\mathcal{L}_{KD}$ is written in Eq. (8)

$$\mathcal{L}_{KD} = \frac{1}{n_{unl}}\sum_{i=1}^{4}\sum_{j=1}^{n_{unl}}||V_j^{wea,i}/t_{KD} - V_j^{tim,i}/t_{KD}||_2^2 \tag{8}$$

where, t_{KD} is a temperature coefficient to scale the features of the intermediate layers, which facilitates the knowledge flow

Algorithm 2 CapMatch

Input: $\mathcal{D} = (\mathcal{D}_{train}, \mathcal{D}_{val}, \mathcal{D}_{test})$;
Output: $\mathcal{Y}$;

Initialize model parameters θ_0;
//Training and validation
for $i = 1$ to n_{ep} **do** $\triangleright$ n_{ep} is the number of training epochs.
    Feedforward $\mathcal{D}_{train}$ into CapMatch;
    Obtain $\mathcal{L}_{KD}$ using Eq. (8);
    Obtain $\mathcal{L}_{CL}$ using Eq. (9);
    Obtain $\mathcal{L}_{sup}$ using Eq. (10);
    Obtain $\mathcal{L}_{CM}$ using Eq. (11);
    Obtain $\mathcal{L}_{uns}$ using Eq. (12);
    Obtain $\mathcal{L}$ using Eq. (13);
    Update θ_i using $\theta_i = \theta_{i-1} - \eta \nabla_{\theta_{i-1}} \mathcal{L}(\theta_{i-1})$; $\triangleright$ η is the learning rate, and θ_{i-1} and $\nabla_{\theta_{i-1}}$ denote the parameters and gradient at the (i-1)-th training epoch, respectively.
    if $i > 1$ **then**
        Validate CapMatch using $\mathcal{D}_{val}$;
    end if
end for
// Testing the model
Use the trained model to predict $\mathcal{Y}$ of $\mathcal{D}_{test}$.

in the intermediate layers. In this paper, we set $t_{KD} = 1.0$ (More details can be found in Section IV.C).

G. Contrastive Learning

As aforementioned, CL distinguishes the similarity between different views from the same sample and that between the views from different samples via a CL loss function, $\mathcal{L}_{CL}$. Let V_j^{wea} and V_j^{tim} be the outputs of the capsule transformer network associated with x_j^{wea} and x_j^{tim}, respectively. As [56], [58], [61] suggest, we define $\mathcal{L}_{CL}$ as:

$$\mathcal{L}_{CL} = -\sum_{j=1}^{n_{unl}} log \frac{exp(sim(V_j^{wea}, V_j^{tim})/t_{CL})}{\sum_{m=1}^{n_{unl}} \mathbb{1}_{[m \neq j]} exp(sim(V_j^{wea}, V_m^{tim})/t_{CL})} \tag{9}$$

where,

$$sim(p, q) = \frac{p^T q}{||p|| ||q||},$$

and t_{CL} is a contrastive coefficient for $\mathcal{L}_{CL}$. This paper sets $t_{CL} = 1.0$ in our experiments (More details can be found in Section IV.C).

H. Loss Function

The loss function, $\mathcal{L}$, includes a supervised loss, $\mathcal{L}_{sup}$, and an unsupervised loss, $\mathcal{L}_{uns}$. As the studies in [34], [35], [36], [37] suggest, $\mathcal{L}_{sup}$ uses the margin loss function to measure the average differences between the ground labels and their prediction vectors on labeled HAR data. $\mathcal{L}_{sup}$ is written as:

$$\begin{aligned}\mathcal{L}_{sup} = \frac{1}{n_{lab}} \sum_{i=1}^{n_{lab}} &(y_i max(0, m^+ - ||V_i^{lab}||) \\ &+ \lambda(1 - y_i) max(0, ||V_i^{lab}|| - m^-))\end{aligned} \tag{10}$$

where, V_i^{lab} is the output of the proposed capsule transformer network associated with the labeled data, x_i^{lab}, and m^+, m^-, and λ are three coefficients for $\mathcal{L}_{sup}$. As the previous studies suggest [34], [35], [37], we set m^+ = 0.9, m^- = 0.1, and λ = 0.5.

$\mathcal{L}_{uns}$ consists of a KD loss function, $\mathcal{L}_{KD}$, a CL loss function, $\mathcal{L}_{CL}$, and a confidential marginal loss function, $\mathcal{L}_{CM}$. Similar to FixMatch [29], CapMatch leverages V_j^{wea} to generate an artificial label associated with V_j^{tim} when $max(V_j^{wea}) \geq t_{CM}$, where t_{CM} is a coefficient for $\mathcal{L}_{CM}$. $\mathcal{L}_{CM}$ is calculated by Eq. (11):

$$\begin{aligned}\mathcal{L}_{CM} = \frac{1}{n_{unl}} \sum_{j=1}^{n_{unl}} &(y_j^{one} max(0, m^+ - ||V_j^{tim}||) \\ &+ \lambda(1 - y_j^{one}) max(0, ||V_j^{tim}|| - m^-))\end{aligned} \tag{11}$$

where,

$$y_j^{one} = \mathbb{1}(max(V_j^{wea}) \geq t_{CM}) argmax(V_j^{wea}),$$

and $argmax(V_j^{wea})$ produces a valid one-hot probability distribution of V_j^{wea}. Following [29], we set t_{CM} = 0.95.

The unsupervised loss of CapMatch, $\mathcal{L}_{uns}$, is defined in Eq. (12).

$$\mathcal{L}_{uns} = \mathcal{L}_{CM} + \mathcal{L}_{KD} + \tau \mathcal{L}_{CL} \tag{12}$$

where, τ is a coefficient of $\mathcal{L}_{CL}$. Following the previous work in [78], we set $\tau = 0.1$.

Thus, the loss function of CapMatch, $\mathcal{L}$, is calculated as:

$$\begin{aligned}\mathcal{L} &= \mathcal{L}_{sup} + \mathcal{L}_{uns} + \epsilon ||\theta||_2^2 \\ &= \mathcal{L}_{sup} + \mathcal{L}_{CM} + \mathcal{L}_{KD} + \tau \mathcal{L}_{CL} + \epsilon ||\theta||_2^2\end{aligned} \tag{13}$$

where, θ represents the model parameters of CapMatch, and ϵ is a coefficient of $||\theta||_2^2$ (i.e., L_2 regularization). Following [37], we set ϵ = 0.0005 in our experiments. Besides, the CapMatch's pseudo-code is given in Algorithm 2.

IV. Experiments

This section first describes the experimental setup, performance metrics, hyper-parameter sensitivity, and ablation study. Then, it verifies the CapMatch's overall performance and computational complexity.

A. Experimental Setting

1) Data Description: To evaluate the performance of CapMatch, we choose three widely used HAR datasets, as follows:

- **HAPT**: the smartphone-based recognition of human activities and postural transitions dataset (HAPT) [79] was collected from 30 volunteers aged 19-48 years. The sensor signals, i.e., accelerometer and gyroscope with noise filters, were set to sample in fixed-width sliding windows of 2.56 sec and 50% overlap (128 readings/window). Each sample is a 561-feature vector with time and frequency domain variables. This dataset consists of six basic activities, i.e., standing, sitting, laying, walking, walking_downstairs and walking_upstairs, and six static postures, including stand-to-sit, sit-to-stand, sit-to-lie, lie-to-sit, stand-to-lie, and lie-to-stand.

TABLE I
DETAILS OF THE THREE HAR DATASETS.

Dataset	Sample Rate	Activities	Classes	Samples	CapMatch's Parameter (M)
HAPT	50Hz	Walking (Wk), Walking_Upstairs (Wu), Walking_Downstairs (Wd), Sitting (St), Standing (Sd), Laying (Ly), Stand-to-Sit (DtS), Sit-to-Stand (StD), Sit-to-Lie (StL), Lie-to-Sit (LtS), Stand-to-Lie(DtL), and Lie-to-Stand (LtD)	12	10,929	2.655756
WISDM	20Hz	Walking (Wk), Jogging (Jg), Upstairs (Us), Downstairs (Ds), Sitting (St), and Standing (Sd)	6	1,098,207	1.865496
UCI_HAR	50Hz	Walking (Wk), Walking_Upstairs (Wu), Walking_Downstairs (Wd), Sitting (St), Standing (Sd), and Laying (Ly)	6	10,299	2.637324

- **WISDM**: the Wireless Sensor Data Mining (WISDM) [80] lab collected the accelerometer data every 50ms, where the signal sample rate was set to 20Hz. It has 1,098,207 multiple physical activities' examples with six attributes: user, activity, timestamp, x-acceleration, y-acceleration, and z-acceleration. This dataset considers six activities, namely, walking, jogging, upstairs, downstairs, sitting, and standing.
- **UCI_HAR**: the human activity recognition using smartphones dataset [81] in the University of California Irvine Machine Learning Repository (UCI_HAR) was collected from 30 volunteers aged 19-48 years. Each volunteer who wore a smartphone (Samsung Galaxy S II) on his/her waist performed six activities: walking, walking_upstairs, walking_downstairs, sitting, standing and laying. The 3-axial linear acceleration and 3-axial angular velocity at a constant rate of 50Hz were used, and the signal sample rate was set to 20Hz.

The summary of the three datasets is shown in Table I.

2) Data Preprocessing: As suggested by [5], [6], [7], [9], [10], this paper adopts the fixed time window method to effectively fuse the activity data collected by different sensors, in which each sensor eliminates the interference caused by noise and obtains the frequency of stable sampling and stable data via filtering technologies, such as low-pass, Kalman, and wavelet filters. Meanwhile, the sequence data collected within a fixed time window can carry much adequate and valuable information about activities and is usually regarded as the input of a HAR algorithm. Let D_i represent the data collected from the sensor at the i-th timestamp, and L be the size of the fixed time window. The sequence data collected in the fixed time window, Seq_j, is defined as:

$$Seq_j = [D_i, D_{i+1}, ..., D_{i+L-1}], \quad j = 1, 2, ..., N_{seq} \tag{14}$$

where, N_{seq} is the number of the sequence data.

3) Data Partition: As the previous studies suggest [5], [6], [7], [44], [45], [46], [47], [48], [49], [50], each given dataset is divided into two parts with a ratio of 7:3. For all supervised and semi-supervised algorithms for comparison, the first part is divided into training and validation sets with a ratio of 8:2, while the second one is regarded as the testing set. For supervised algorithms, the data partition process is over. For semi-supervised algorithms, the training set of the first part is further split into labeled and unlabeled data, where unlabeled data are those with labels removed. Like [82], [83], [84], the ratio of labeled data to the whole training data, $r = \frac{n_{lab}}{n_{lab}+n_{unl}}$ (s.t., $n_{lab} \ll n_{unl}$), is from 0.1 to 0.3, i.e., $r \in \{0.1, 0.2, 0.3\}$. On the other hand, all semi-supervised and supervised algorithms are verified by the same testing set.

TABLE II
HYPER-PARAMETER SETTINGS OF THE FOUR CAPSULE-BASED TRANSFORMER BLOCKS.

Transformer No.	Fully-connected layer's units	n_{att}	Dropout Value
1	48	8	0.5
2	96	8	0.5
3	144	8	0.5
4	192	8	0.5

4) Implementation details: The hyper-parameter settings of the four capsule-based transformer blocks are given in Table II. This paper uses RMSPropOptimizer as the optimizer, with the momentum term, initial learning rate, and decay value set to 0.9, 0.001, and 0.9, respectively. We conduct the experiments with a computer with Ubuntu 18.04 OS, an Nvidia GTX 1080Ti GPU with 11GB, and an AMD R5 1400 CPU with 16G RAM.

B. Performance Metrics

As suggested in [5], [6], [7], [12], [44], [45], we use two commonly used metrics, i.e., $Accuracy$ and F-measure, in performance comparison. These metrics are defined as:

$$\begin{aligned} Accuracy &= \frac{tp+tn}{tp+tn+fp+fn} \times 100\% \\ F_1 &= \frac{Precision \times Recall}{Precision + Recall} \\ Precision &= \frac{tp}{tp+fn} \times 100\% \\ Recall &= \frac{tp}{tp+tn} \times 100\% \end{aligned} \tag{15}$$

where, tp and tn are the numbers of true positive and negative samples, respectively. fp and fn represent the numbers of false positive and negative samples, respectively.

C. Hyper-parameter Sensitivity

We study the influence of hyper-parameter settings on the performance of CapMatch on the HAPT, WISDM, and UCI_HAR datasets.

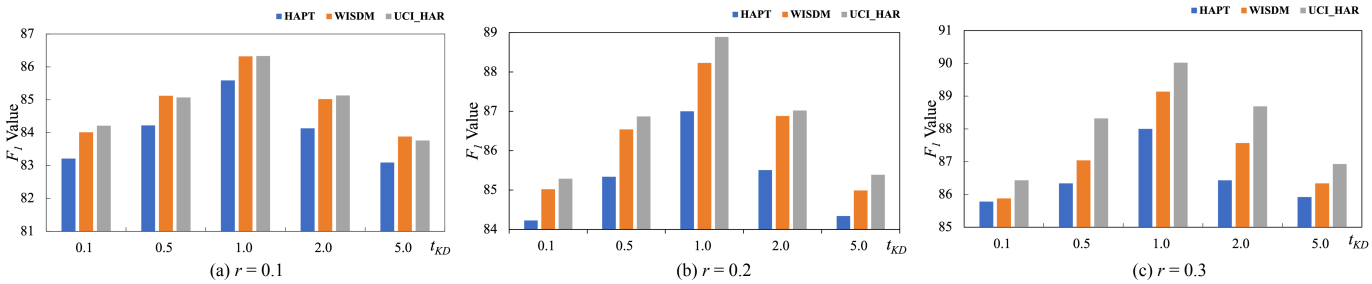


Fig. 4. F_1 results with different t_{KD} values when r = 0.1, 0.2, and 0.3.

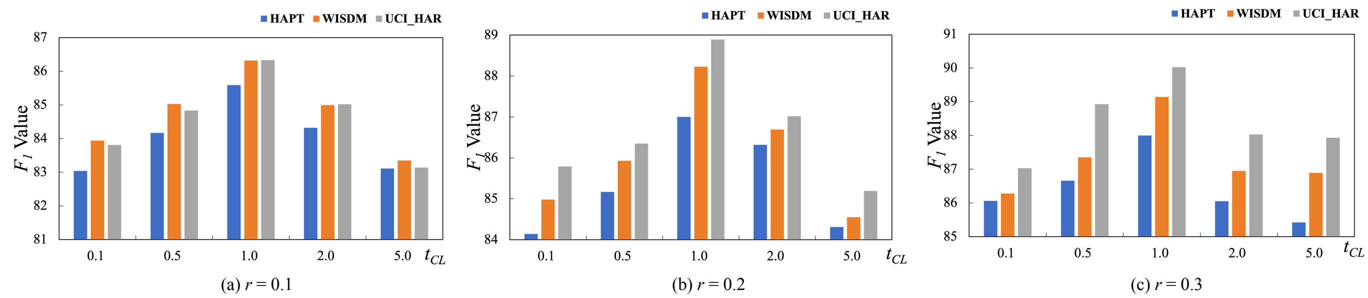


Fig. 5. F_1 results with different t_{CL} values when r = 0.1, 0.2, and 0.3.

1) CapMatch with different t_{KD} values: t_{KD} is a temperature coefficient to scale the features of the intermediate layers, which facilitates the knowledge flow in the intermediate layers. As shown in Figure 4, 1.0 is the best setting for t_{KD} because it helps CapMatch to obtain the highest F_1 value on each HAR dataset.

2) CapMatch with different t_{CL} values: t_{CL} is a threshold value for CapMatch to learn the similarity between different views from the same sample. Figure 5 shows the F_1 results obtained by CapMatch with different t_{CL} (i.e., $t_{CL} \in \{0.1, 0.5, 1.0, 2.0, 5.0\}$) values on three HAR datasets when r = 0.1, 0.2, and 0.3. When t_{CL} = 1.0, CapMatch achieves the highest F_1 result on each dataset. That means t_{CL} = 1.0 helps CapMatch mine rich connections and regularizations from the HAR data.

TABLE III

F_1 RESULTS OBTAINED BY CAPMATCH WITH DIFFERENT KD LOSSES WHEN r = 0.1, 0.2, AND 0.3. ABBREVIATIONS: L_1 – L_1 LOSS, CE – CROSS ENTROPY, L_2 – L_2 LOSS.

r	Dataset	L_1	L_2	KL	CE
0.1	HAPT	85.04 ± 1.15	85.59 ± 1.06	85.42 ± 1.12	84.69 ± 1.04
	WISDM	84.94 ± 0.79	86.32 ± 0.82	84.99 ± 0.85	84.52 ± 0.86
	UCI_HAR	85.81 ± 1.58	86.33 ± 1.73	85.32 ± 1.58	85.98 ± 1.65
0.2	HAPT	85.89 ± 1.24	87.00 ± 1.01	86.01 ± 1.13	85.59 ± 1.23
	WISDM	86.83 ± 0.52	88.23 ± 0.48	87.17 ± 0.56	85.36 ± 0.63
	UCI_HAR	87.03 ± 1.26	88.89 ± 1.36	87.84 ± 1.28	86.86 ± 1.16
0.3	HAPT	86.58 ± 1.19	**88.00** ± 1.03	86.59 ± 1.18	85.89 ± 1.24
	WISDM	87.22 ± 0.68	**89.14** ± 0.89	88.02 ± 0.69	87.03 ± 0.58
	UCI_HAR	88.57 ± 1.28	**90.02** ± 1.21	88.94 ± 1.25	87.84 ± 1.28

3) CapMatch with different KD losses: It is crucial to choose an appropriate KD loss function to measure the knowledge difference between a teacher and its student. Table III shows the F_1 results obtained by CapMatch with different KD losses on three HAR datasets when r = 0.1, 0.2, and 0.3. L_2 performs better than the other 3 losses. Hence, we choose the L_2 loss to promote the knowledge transfer within the model.

TABLE IV

F_1 RESULTS OBTAINED BY SEVEN CAPMATCH VARIANTS WHEN r = 0.1, 0.2, AND 0.3.

r	Method	HAPT	WISDM	UCI_HAR
0.1	CapMatch w/o Routing	78.23 ± 1.13	75.05 ± 0.79	76.13 ± 1.62
	CapMatch w/o CL	84.22 ± 1.12	84.15 ± 0.76	84.33 ± 1.58
	CapMatch w/o FKD	83.98 ± 1.16	83.29 ± 0.74	83.53 ± 1.57
	CapMatch w RKD	84.79 ± 1.24	84.79 ± 0.68	84.85 ± 1.26
	VanMatch	84.37 ± 1.04	84.68 ± 0.73	84.97 ± 1.35
	EmbMatch	84.62 ± 1.17	85.03 ± 0.59	85.02 ± 1.25
	CapMatch	85.59 ± 1.06	86.32 ± 0.82	86.33 ± 1.73
0.2	CapMatch w/o Routing	81.21 ± 1.35	81.99 ± 0.46	82.56 ± 1.45
	CapMatch w/o CL	84.85 ± 1.14	84.92 ± 0.48	85.76 ± 1.22
	CapMatch w/o FKD	85.03 ± 1.16	85.00 ± 0.53	85.97 ± 1.17
	CapMatch w RKD	86.03 ± 1.24	86.49 ± 0.62	86.33 ± 1.73
	VanMatch	86.45 ± 1.17	87.03 ± 0.78	87.92 ± 1.25
	EmbMatch	85.72 ± 1.29	86.92 ± 0.63	86.29 ± 1.27
	CapMatch	87.00 ± 1.01	88.23 ± 0.48	88.89 ± 1.36
0.3	CapMatch w/o Routing	82.88 ± 1.17	82.93 ± 0.56	83.75 ± 1.15
	CapMatch w/o CL	85.03 ± 1.16	84.02 ± 0.56	85.95 ± 1.19
	CapMatch w/o FKD	86.39 ± 1.32	86.92 ± 0.63	88.05 ± 1.23
	CapMatch w RKD	86.89 ± 1.27	87.94 ± 0.56	88.26 ± 1.28
	VanMatch	87.00 ± 1.01	88.25 ± 0.57	88.92 ± 1.34
	EmbMatch	87.34 ± 1.08	88.69 ± 0.68	89.17 ± 1.17
	CapMatch	**88.00** ± 1.03	**89.14** ± 0.89	**90.02** ± 1.21

D. Ablation Study

We investigate the effects of different components on CapMatch on three HAR datasets.

1) Effectiveness of Routing: To verify the contribution of routing on CapMatch, we compare it with CapMatch without the routing mechanism (called *CapMatch w/o Routing*) on three HAR datasets. Table IV shows the F_1 results obtained by different CapMatch variants on three HAR datasets when r = 0.1, 0.2, and 0.3. One can easily see that the routing mechanism significantly improves the performance of CapMatch on F_1. That is why CapMatch outperforms *CapMatch w/o Routing* on each dataset.

2) Effectiveness of Contrastive Learning: To study the impact of CL on CapMatch, we compare it with CapMatch without contrastive learning (called *CapMatch w/o CL*) on three HAR datasets. As shown in table IV, CapMatch results in a higher F_1 value on each dataset since CL helps enhance the quality of the representations learned by quantifying the differences between different views of the same sample.

3) Effectiveness of Feature-based Knowledge Distillation: To explore the effects of the feature-based KD on CapMatch, we compare it with two variants on three HAR datasets, listed below.

- *CapMatch w/o FKD*: CapMatch without feature-based KD.
- *CapMatch w RKD*: CapMatch with response-based KD [32].

One can observe that CapMatch overweighs *CapMatch w/o FKD* and *CapMatch w/o RKD* in terms of F_1 measure because the feature-based KD improves knowledge transfer between intermediate layers of the model.

4) Effectiveness of Capsule-based Transformer: To investigate the effects of capsule-based transformer on CapMatch, we compare it with two variants:

- *VanMatch*: CapMatch with each capsule-based transformer block replaced with the vanilla transformer block [74], [75].
- *EmbMatch*: CapMatch with each capsule-based transformer block replaced with the embedding transformer block [76].

As shown in table IV, CapMatch outperforms VanMatch and EmbMatch on each dataset because the capsule-based transformer block relates the capsules at different locations, being able to mine rich connections and regulations hidden in HAR data.

In summary, routing, CL, feature-based KD, and capsule-based transformer are all essential components for CapMatch.

E. Experimental Analysis

To evaluate the performance of CapMatch, we compare it with 14 SSL algorithms against F_1 value, as follows:

- *DTW-D*: a modified dynamic time warping algorithm for HAR [84].
- *Self-labeled SVM*: a modified self-labeled algorithm with SVM for HAR [85].
- *Self-labeled Clustering*: a modified self-labeled algorithm with clustering for HAR [85].
- *SSSL*: a combination of the shapelet method and pseudo-labeling for HAR [83].
- *SelfHAR*: a self-supervised learning algorithm for HAR [24].
- *SSRCA*: a semi-supervised recurrent convolutional attention algorithm for HAR [19].
- *UDA*: based on data augmentation and consistency regularization with the proposed capsule transformer network (see Figure 1) as its feature extractor [86].
- *En-Co-Training*: an SSL algorithm with co-training for HAR [20].
- *Sparse-Coding*: a sparse-coding SSL framework for HAR [87].
- *SSCLHAR*: a contrastive SSL algorithm for HAR [25].
- *ActSemiCNN*: an active semi-supervised CNN for HAR [22].
- *MixMatch*: a modified MixMatch [27] adapted to HAR, with the proposed capsule transformer network (see Figure 1) as its feature extractor.
- *FixMatch*: a modified FixMatch [29] adapted to HAR, with the proposed capsule transformer network (see Figure 1) as its feature extractor.
- *FlexMatch*: a modified FlexMatch [30] adapted to HAR, with the proposed capsule transformer network (see Figure 1) as its feature extractor.

Table V shows the F_1 results with various SSL algorithms with different r values on three HAR datasets. One can easily observe that CapMatch performs the best among all compared SSL algorithms on each dataset, e.g., CapMatch obtains the highest F_1 value on the WISDM dataset when r = 0.1, namely 86.32%. FlexMatch takes the second position while DTW-D leads to the worst performance. The F_1 value of CapMatch is at least 1.3% higher than that of FlexMatch in average with three datasets considered.

The following explains our observations above. Based on the capsule transformer structure, CapMatch gracefully hybridizes pseudo-labeling, CL, and feature-based KD, being able to capture as many intrinsic connections among the obtained representations of classes in the unlabeled data as possible. Thanks to the consistency regularization and curriculum pseudo-labeling techniques, FlexMatch can mine valuable information from the unlabeled data and achieves decent performance regarding F_1 value on three HAR datasets. On the other hand, DTW-D cannot explore rich features and regularizations from the unlabeled data via the DTW technique only.

Second, to study the impact of r on the performance of CapMatch, this paper shows the F_1 results obtained by CapMatch with r = 0.1, 0.2, and 0.3 on three HAR datasets in Figure 6. With more labeled data, more additional prior knowledge is brought to CapMatch, helping it mine richer relationships and regularizations from the data. That is why the CapMatch's performance is gradually enhanced as the amount of labeled data increases.

Finally, we compare CapMatch with r = 0.3 with 16 supervised algorithms on three HAR datasets and collect the F_1 results in Table VI. These supervised algorithms can be classified into traditional and deep learning algorithms, listed below.

- *Traditional Algorithm*:

TABLE V
F_1 RESULTS OBTAINED BY VARIOUS SSL ALGORITHMS WITH DIFFERENT r VALUES ON THREE HAR DATASETS.

r	Author	Method	HAPT (%)	WISDM (%)	UCI_HAR (%)
0.1	Chen *et al.* [84]	DTW-D	73.55 ± 1.41	74.25 ± 0.59	74.68 ± 1.45
	Zhou *et al.* [85]	Self-labeled SVM	78.35 ± 1.29	77.25 ± 0.78	76.40 ± 1.52
		Self-labeled Clustering	81.08 ± 1.23	75.34 ± 0.36	79.32 ± 1.03
	Han *et al.* [83]	SSSL	80.95 ± 1.27	80.92 ± 0.52	81.95 ± 1.47
	Tang *et al.* [24]	SelfHAR	83.71 ± 1.15	82.29 ± 0.65	82.02 ± 1.22
	Chen *et al.* [19]	SSRCA	—	—	81.43
	Xie *et al.* [86]	UDA	81.21 ± 1.35	80.93 ± 0.73	81.94 ± 1.26
	Guan *et al.* [20]	En-Co-Training	75.15 ± 1.28	77.12 ± 0.58	79.26 ± 2.04
	Bhattacharya *et al.* [87]	Sparse-Coding	76.65 ± 1.25	75.93 ± 0.68	76.20 ± 2.15
	Khaertdinov *et al.* [25]	SSCLHAR	82.74 ± 1.09	81.99 ± 0.46	83.32 ± 1.42
	Bi *et al.* [22]	ActSemiCNN	83.54 ± 1.12	83.19 ± 0.59	82.99 ± 1.39
	Berthelot *et al.* [27]	MixMatch	82.02 ± 1.21	82.72 ± 0.73	83.24 ± 1.06
	Sohn *et al.* [29]	FixMatch	83.25 ± 1.13	84.02 ± 0.56	84.71 ± 1.18
	Zhang *et al.* [30]	FlexMatch	83.98 ± 1.21	84.12 ± 0.61	84.99 ± 1.01
	Ours	CapMatch	85.59 ± 1.06	86.32 ± 0.82	86.33 ± 1.73
0.2	Chen *et al.* [84]	DTW-D	74.05 ± 1.39	75.07 ± 0.64	75.23 ± 1.83
	Zhou *et al.* [85]	Self-labeled SVM	79.11 ± 1.18	78.81 ± 0.59	79.93 ± 1.16
		Self-labeled Clustering	81.94 ± 1.22	78.33 ± 0.49	80.01 ± 1.20
	Han *et al.* [83]	SSSL	81.44 ± 1.58	81.79 ± 0.55	82.56 ± 1.45
	Tang *et al.* [24]	SelfHAR	84.23 ± 1.29	83.11 ± 0.58	83.01 ± 1.14
	Chen *et al.* [19]	SSRCA	—	—	—
	Xie *et al.* [86]	UDA	81.99 ± 1.19	81.54 ± 0.49	82.88 ± 1.35
	Guan *et al.* [20]	En-Co-Training	76.23 ± 1.23	76.97 ± 0.58	77.09 ± 1.25
	Bhattacharya *et al.* [87]	Sparse-Coding	77.22 ± 1.19	76.69 ± 0.62	77.99 ± 1.19
	Khaertdinov *et al.* [25]	SSCLHAR	83.39 ± 1.24	82.89 ± 0.54	84.68 ± 1.21
	Bi *et al.* [22]	ActSemiCNN	84.97 ± 1.32	84.87 ± 0.52	84.03 ± 1.22
	Berthelot *et al.* [27]	MixMatch	83.44 ± 1.20	83.93 ± 0.49	84.87 ± 1.17
	Sohn *et al.* [29]	FixMatch	84.95 ± 1.18	85.92 ± 0.53	85.76 ± 1.22
	Zhang *et al.* [30]	FlexMatch	85.39 ± 1.26	86.52 ± 0.53	86.32 ± 1.08
	Ours	CapMatch	87.00 ± 1.01	88.23 ± 0.48	88.89 ± 1.36
0.3	Chen *et al.* [84]	DTW-D	75.92 ± 1.22	76.12 ± 0.49	76.22 ± 1.19
	Zhou *et al.* [85]	Self-labeled SVM	80.01 ± 1.12	79.94 ± 0.71	80.59 ± 1.23
		Self-labeled Clustering	82.77 ± 1.20	79.83 ± 0.48	80.93 ± 1.19
	Han *et al.* [83]	SSSL	82.97 ± 1.15	82.93 ± 0.66	83.11 ± 1.17
	Tang *et al.* [24]	SelfHAR	85.02 ± 1.08	84.44 ± 0.52	83.75 ± 1.15
	Chen *et al.* [19]	SSRCA	—	—	—
	Xie *et al.* [86]	UDA	82.88 ± 1.17	82.22 ± 0.68	83.33 ± 1.19
	Guan *et al.* [20]	En-Co-Training	77.58 ± 1.18	77.59 ± 0.53	77.96 ± 1.24
	Bhattacharya *et al.* [87]	Sparse-Coding	79.01 ± 1.07	78.09 ± 0.65	79.25 ± 1.17
	Khaertdinov *et al.* [25]	SSCLHAR	84.43 ± 1.13	83.99 ± 0.53	85.05 ± 1.19
	Bi *et al.* [22]	ActSemiCNN	85.83 ± 1.18	86.03 ± 0.60	85.83 ± 1.09
	Berthelot *et al.* [27]	MixMatch	85.03 ± 1.16	84.92 ± 0.48	85.97 ± 1.17
	Sohn *et al.* [29]	FixMatch	86.02 ± 1.07	86.97 ± 0.35	87.13 ± 1.18
	Zhang *et al.* [30]	FlexMatch	86.35 ± 1.18	87.33 ± 0.54	88.19 ± 1.05
	Ours	CapMatch	**88.00** ± 1.03	**89.14** ± 0.89	**90.02** ± 1.21

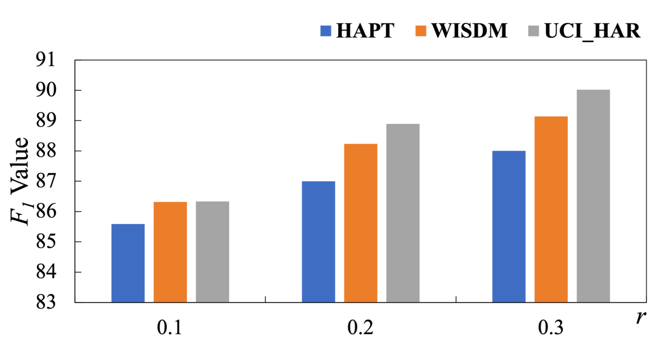


Fig. 6. F_1 results obtained by CapMatch with r = 0.1, 0.2, and 0.3 on three datasets.

- primary machine learning algorithms: SVM, K-Nearest Neighbor (KNN), GradientBoosting, Random Forest, and Decision Tree [5].
- *J48*: a machine learning algorithm based on decision tree using iterative Dichotomiser [7].

- *Deep Learning Algorithm*
  - *Stacked Denoising Autoencoder*: a stacked denoising autoencoder method based fully-connected neural networks for HAR [45].
  - *1DCNN*: a one-dimensional CNN model for HAR [44].
  - *2DCNN*: a two-dimensional CNN model for HAR [44].
  - *Multi-head Convolutional Attention*: a multi-head feature network integrating multi-head CNN and attention for HAR [44].
  - *CNNLSTM*: a cascading model based on CNN and LSTM for HAR [51].

TABLE VI

F_1 RESULTS OBTAINED BY CAPMATCH WITH r = 0.3 AND 16 EXISTING SUPERVISED ALGORITHMS ON THREE HAR DATASETS.

Training Scheme	Method	HAPT (%)	WISDM (%)	UCI_HAR (%)
Semi-superved Learning	CapMatch with r = 0.3	88.00 ± 1.03	89.14 ± 0.89	90.02 ± 1.21
Supervised Learning (Traditional Algorithms)	SVM	94.14	88.26	94.00
	KNN	87.01	82.29	90.00
	GradientBoosting	90.00	79.71	93.90
	Random Forest	83.74	82.38	90.17
	Decision Tree	72.52	73.17	85.86
	J48	–	85.21	–
Supervised Learning (Deep Learning Algorithms)	Stacked Denoising Autoencoder	–	94.01	–
	1DCNN	91.15	92.13	91.72
	2DCNN	90.98	91.89	92.60
	Multi-head Convolutional Attention	94.79	95.68	95.40
	CNNLSTM	93.15	94.92	94.89
	CNNBiLSTM	94.02	95.83	95.37
	LegoCNN	94.59	97.31	95.41
	Perceptive Extraction Network	95.31	98.97	**96.33**
	Deformable CNN	–	**99.21**	–
	Selective Kernel Convolution	**96.11**	98.13	–

TABLE VII

THE NUMBER OF PARAMETERS AND RUN TIME RESULTS WITH VARIOUS ALGORITHMS ON THREE HAR TESTING DATASETS.

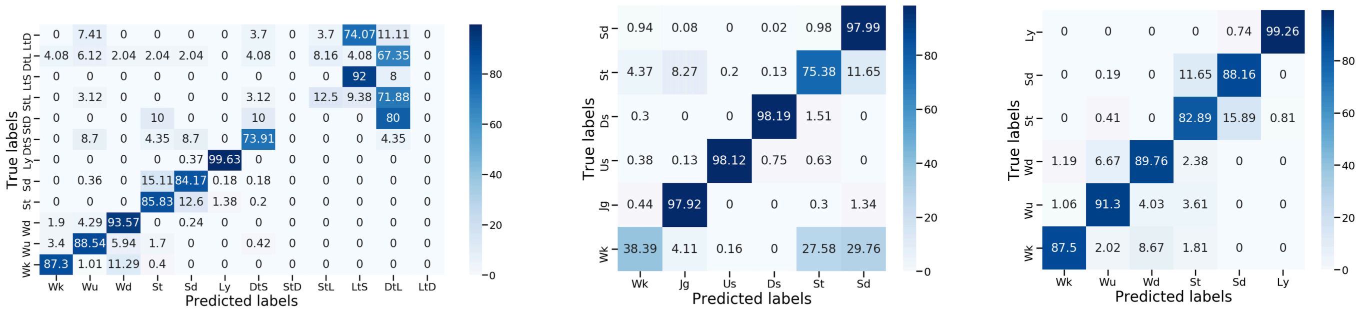
Method	HAPT			WISDM			UCI_HAR		
	Parameters (M)	With CPU (s)	With GPU (s)	Parameters (M)	With CPU (s)	With GPU (s)	Parameters (M)	With CPU (s)	With GPU (s)
SVM	—	**0.2974**	—	—	**5.8365**	—	—	**0.2059**	—
KNN	—	1.2254	—	—	9.4432	—	—	0.9346	—
BAGGING	—	1.8502	—	—	6.2953	—	—	1.1045	—
Random Forest	—	3.6834	—	—	9.2659	—	—	3.4596	—
Multilayer Perceptron	**0.167468**	1.4246	**0.2123**	**0.089638**	7.3983	**1.1658**	**0.164396**	1.0758	**0.1964**
1DCNN	0.923217	16.2392	2.2394	0.772960	19.0011	8.2335	0.920145	15.2322	2.1262
2DCNN	1.158791	33.8867	3.0015	1.124334	21.7324	7.8746	1.155719	31.6259	2.8849
LSTM	1.367559	5.2358	1.4599	0.204324	16.8543	5.0002	1.364487	3.1376	0.9435
CNNLSTM	2.916935	38.8235	8.2305	2.837317	69.6823	17.2398	2.913863	37.7875	7.0921
Multi-head Convolutional Attention	2.899985	36.9934	6.9611	2.771270	22.3498	11.8345	2.896913	35.8435	5.8934
Perceptive Extraction Network	0.822983	9.0021	1.6789	0.223558	17.9467	7.4801	0.819911	7.4029	1.4934
Deformable CNN	—	—	—	6.640000	—	—	—	—	—
Adaptive Deep Network	—	—	—	5.591000	—	—	—	—	—
Selective Kernel Convolution	—	—	—	0.360000	—	—	0.45	—	—
Shallow CNNs	—	—	—	—	—	—	0.341	—	—
CapMatch	2.655756	34.0248	5.1223	1.865496	21.8735	9.4529	2.637324	32.3743	4.9345

– *CNNBiLSTM*: a cascading model based on CNN and BiLSTM for HAR [51].
– *LegoCNN*: a lightweight CNN model based on Lego filters for HAR [47].
– *Perceptive Extraction Network*: a perceptive extraction network integrating a feature network and a relation network in parallel for HAR [6].
– *Deformable CNN*: a learning model based on deformable CNN for HAR [48].
– *Selective Kernel Convolution*: a multi-branch CNN model based on selective kernel convolution for HAR [50].

All semi-supervised and supervised algorithms are verified by the same testing datasets. One can easily see that CapMatch performs worse than all deep learning-based supervised algorithms but it performs better than several traditional supervised algorithms. To be specific, CapMatch outperforms KNN, Random Forest, and Decision Tree on HAPT, performs the best on WISDM, and performs better than Decision Tree on UCR_HAR. For example, the F_1 value of CapMatch is 89.14% while that of KNN is 82.29% on the WISDM dataset. Therefore, CapMatch has the potential to address various SSL tasks in the HAR domain. Besides, to visualize the performance of CapMatch with r = 0.3 for each class of activity on each dataset, we show its confusion matrices on three datasets in Figure 7. One can see that CapMatch with r = 0.3 performs the best on the UCI_HAR dataset, as the accuracy of each class activity exceeds 80%. On the HAPT dataset, CapMatch with r = 0.3 performs the worst because it cannot mine rich features from the short event data with few labels, resulting in poor performance in short events, e.g., Lie-to-Stand (Ltd) and Sit-to-Lie (StL). Overall, CapMatch with r = 0.3 performs well in most of the activity categories on three datasets, demonstrating its excellent feature extraction ability.

F. Computational Complexity

To evaluate the efficiency of CapMatch, we compare it with a number of machine and deep learning algorithms regarding the number of parameters and run time on three HAR testing datasets, as shown in Table VII. One can easily observe that CapMatch is slower than 4 machine learning algorithms,



(a) HAPT (b) WISDM (c) UCI_HAR

Fig. 7. Confusion matrixes of CapMatch with r = 0.3 on three HAR datasets.

including SVM, BAGGING, Random Forest, and KNN. When compared with deep learning algorithms, CapMatch is faster than CNNLSTM and Multi-head Convolutional Attention, but slower than the others.

V. Conclusion

CapMatch gracefully integrates supervised learning and unsupervised learning into the proposed capsule transformer network, being able to extract abundant representations from partially labeled HAR data, where pseudo-labeling, contrastive learning, and feature-based knowledge distillation are adopted to establish similarity learning on the lower- and higher-level semantic information extracted. As the feature extractor of CapMatch, the capsule transformer network can capture sufficient local and global patterns of HAR data. With 10%, 20%, and 30% of data labeled, CapMatch performs the best among all compared semi-supervised algorithms on the HAPT, WISDM, and UCI_HAR datasets. With 30% of data labeled, CapMatch performs even better than a number of classical supervised algorithms, achieving an F_1 value of 88.00% on HAPT, 89.14% on WISDM, and 90.02% on UCI_HAR. In particular, on the WISDM dataset, CapMatch outperforms all classical supervised algorithms for comparison, including SVM, KNN, GradientBoosting, Random Forest, Decision Tree, and J48. That reflects the potential of CapMatch to be applied to various real-world HAR problems.

CapMatch is not well suited for direct deployment on lightweight devices to handle real-time HAR tasks. In the future, we will consider introducing network pruning techniques into CapMatch to build a lightweight CapMath for real-time HAR tasks.

Acknowledgement

The authors genuinely thank the editors and reviewers for providing many constructive suggestions to enhance the quality of the article.

References

[1] D. Anguita, L. O. A. Ghio, X. Parra, and J. Reyes-Ortiz, “A public domain dataset for human activity recognition using smartphones,” in *Proc. 21st Eur. Symposium Artif. Neural Netw., Comput. Intell. Mach. Learn.*, 2013, pp. 437–442.

[2] Y. Zhang, T. Zhou, W. Wu, H. Xie, H. Zhu, G. Zhou, and A. Cichocki, “Improving eeg decoding via clustering-based multitask feature learning,” *IEEE Trans. Neur. Net. Lear.*, vol. 33, no. 8, pp. 3587–3597, 2022.

[3] Q. Li, Z. Luo, and J. Zheng, “A new deep anomaly detection-based method for user authentication using multichannel surface emg signals of hand gestures,” *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–11, 2022.

[4] Q. Zhai, X. Han, Y. Han, J. Yi, S. Wang, and T. Liu, “A contactless on-bed radar system for human respiration monitoring,” *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–10, 2022.

[5] F. Serpush, M. B. Menhaj, B. Masoumi, and B. Karasfi, “Wearable sensor-based human activity recognition in the smart healthcare system,” *Comput. Intel. Neurosc.*, pp. 1–31, 2022.

[6] Z. Xiao, X. Xu, H. Xing, F. Song, X. Wang, and B. Zhao, “A federated learning system with enhanced feature extraction for human activity recognition,” *Knowl.-Based Syst.*, vol. 229, pp. 1–14, 2021.

[7] Y. Wang, S. Cang, and H. Yu, “A survey on wearable sensor modality centred human activity recognition in health care,” *Expert Syst. Appl.*, vol. 127, pp. 167–190, 2018.

[8] H. Xing, Z. Xiao, R. Qu, Z. Zhu, and B. Zhao, “An efficient federated distillation learning system for multitask time series classification,” *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–12, 2022.

[9] C. Zhu and W. Sheng, “Wearable sensor-based hand gesture and daily activity recognition for robot-assisted living,” *IEEE Trans. Syst. Man Cybern. Part A Syst. Humans*, vol. 41, no. 3, pp. 569–573, 2011.

[10] Z. Chen, Q. Zhu, Y. C. Soh, and L. Zhang, “Robust human activity recognition using smartphone sensors via ct-pca and online svm,” *IEEE Trans. Ind. Inform.*, vol. 13, no. 6, pp. 3070–3080, 2017.

[11] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, pp. 436–444, 2015.

[12] M. A. A. Al-qaness, A. Dahou, M. A. Elaziz, and A. M. Helmi, “Multi-resatt: Multilevel residual network with attention for human activity recognition using wearable sensors,” *IEEE Trans. Ind. Inform.*, vol. 19, no. 1, pp. 144–152, 2023.

[13] S. Xia, L. Chu, L. Pei, Z. Zhang, W. Yu, and R. C. Qiu, “Learning disentangled representation for mixed-reality human activity recognition with a single imu sensor,” *IEEE Trans. Neur. Net. Lear.*, vol. 70, pp. 1–14, 2021.

[14] X. Shu, L. Zhang, Y. Sun, and J. Tang, “Host–parasite: Graph lstm-in-lstm for group activity recognition,” *IEEE Trans. Neur. Net. Lear.*, vol. 32, no. 2, pp. 663–674, 2021.

[15] J. van Engelen and H. H. Hoos, “A survey on semi-supervised learning,” *Mach. Learn.*, vol. 109, pp. 373–440, 2020.

[16] M. Stikic, D. Larlus, and B. Schiele, “Multi-graph based semi-supervised learning for activity recognition,” in *Proc. IEEE Int. Symp. Wearable Comput. (ISWC)*, 2009, pp. 85–92.

[17] L. Yao, F. Nie, Q. Z. Sheng, T. Gu, X. Li, and S. Wang, “Learning from less for better: Semi-supervised activity recognition via shared structure discovery,” in *Proc. ACM Int. Jt. Conf. Pervasive Ubiquitous Comput.*, 2016, pp. 13–24.

[18] W. Liu, S. Fu, Y. Zhou, Z. Zha, and L. Nie, “Human activity recognition by manifold regularization based dynamic graph convolutional networks,” *Neurocomputing*, vol. 444, pp. 217–225, 2021.

[19] K. Chen, L. Yao, D. Zhang, X. Wang, X. Chang, and F. Nie, “A semisupervised recurrent convolutional attention model for human ac-

tivity recognition," *IEEE Trans. Neur. Net. Lear.*, vol. 31, no. 5, pp. 1747–1756, 2020.

[20] D. Guan, W. Yuan, Y.-K. Lee, A. Gavrilov, and S. Lee, "Activity recognition based on semi-supervised learning," in *Proc. Proc. IEEE Int. Conf. Emdedded Real-Time Comput. Syst. Appl.*, 2007, pp. 469–475.

[21] M. Stikic, K. V. Laerhoven, and B. Schiele, "Exploring semi-supervised and active learning for activity recognition," in *Proc. IEEE Int. Symp. Wearable Comput. (ISWC)*, 2008, pp. 81–88.

[22] H. Bi, M. Perello-Nieto, P. Santos-Rodriguez, P. Flach, and I. Craddock, "An active semi-supervised deep learning model for human activity recognition," *J. Amb. Intel. Hum. Comp.*, 2022.

[23] H. Qian, S. J. Pan, and C. Miao, "Distribution-based semi-supervised learning for activity recognition," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 7699–7706.

[24] C. Tang, I. Perez-Pozuelo *et al.*, "Selfhar: Improving human activity recognition through self-training with unlabeled data," in *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 2021, pp. 1–30.

[25] B. Khaertdinov, E. Ghaleb, and S. Asteriadis, "Contrastive self-supervised learning for sensor-based human activity recognition," in *Proc. IEEE Int. Jt. Conf. Biom.*, 2021, pp. 1–8.

[26] Y. Jain *et al.*, "Collossl: Collaborative self-supervised learning for human activity recognition," in *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 2022, pp. 1–28.

[27] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," in *Proc. Adv. Neural Inf. Proces. Syst.*, 2019, pp. 5050–5060.

[28] D. Berthelot, N. Carlini, E. D. Cubuk, A. Kurakin, H. Zhang, and C. Raffel, "Remixmatch: Semi-supervised learning with distribution alignment and augmentation," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–13.

[29] K. Sohn, D. Berthelot, C. Li, Z. Zhang, N. Carlini, E. Cubuk, A. Kurakin, H. Zhang, and C. Raffel, "Fixmatch: Simplifying semi-supervised learning with consistency and confidence," in *Proc. Adv. Neural Inf. Proces. Syst.*, 2020, pp. 1–12.

[30] B. Zhang, Y. Wang, W. Hou, H. Wu, J. Wang, M. Okumura, and T. Shinozaki, "Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling," in *Proc. Adv. Neural Inf. Proces. Syst.*, 2021, pp. 1–12.

[31] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal.*, vol. 35, no. 8, pp. 1798–1828, 2013.

[32] J. Gou, B. Yu *et al.*, "Knowledge distillation: A survey," *Int. J. Comput. Vis.*, vol. 129, pp. 1789–1819, 2021.

[33] L. Jing and Y. Tian, "Self-supervised visual feature learning with deep neural networks: A survey," *IEEE Trans. Pattern Anal. Intell.*, vol. 43, no. 11, pp. 4037–4058, 2021.

[34] S. Sabour, N. Frosst, and G. Hinton, "Dynamic routing between capsules," in *Proc. Adv. Neural Inf. Proces. Syst.*, 2017, pp. 3857–3867.

[35] L. Chen, N. Qin, X. Dai, and D. Huang, "Fault diagnosis of high-speed train bogie based on capsule network," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 6, pp. 6203–6211, 2020.

[36] Y. Feng, J. Gao, and C. Xu, "Learning dual-routing capsule graph neural network for few-shot video classification," *IEEE Trans. Multimedia*, vol. 25, pp. 3204–3216, 2023.

[37] Z. Xiao, X. Xu, H. Zhang, and E. Szczerbicki, "A new multi-process collaborative architecture for time series classification," *Knowl.-Based Syst.*, vol. 220, pp. 1–11, 2021.

[38] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, Ł. Kariser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Proces. Syst.*, 2017, pp. 5998–6008.

[39] A. Gupta, A. Kembhavi, and L. S. Davis, "Observing human-object interactions: Using spatial and functional compatibility for recognition," *IEEE Trans. Pattern Anal.*, vol. 31, no. 10, pp. 1775–1789, 2009.

[40] S. Xia, L. Chu, L. Pei, Z. Zhang, W. Yu, and R. C. Qiu, "Learning disentangled representation for mixed- reality human activity recognition with a single imu sensor," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–14, 2021.

[41] D. Tao, L. Jin, Y. Wang, and X. L, "Rank preserving discriminant analysis for human behavior recognition on wireless sensor networks," *IEEE Trans. Ind. Inform.*, vol. 10, no. 1, pp. 813–823, 2014.

[42] R. A. Hamad, A. S. Hidalgo, M. R. Bouguelia, M. E. Estevez, and J. M. Quero, "Efficient activity recognition in smart homes using delayed fuzzy temporal windows on binary sensors," *IEEE J. Biomed. Health*, vol. 24, no. 2, pp. 387–395, 2020.

[43] D. Ravi, C. Wong, B. Lo, and G. Yang, "Deep learning for human activity recognition: a resource efficient implementation on low-power devices," in *In Proc. Annual Body Sens. Netw. Conf.*, 2016, pp. 71–76.

[44] H. Zhang, Z. Xiao, J. Wang, F. Li, and E. Szczerbicki, "A novel iot-perceptive human activity recognition (har) approach using multihead convolutional attention," *IEEE Internet Things J.*, vol. 7, no. 2, pp. 1072–1080, 2020.

[45] F. Gu, K. Khoshelham, S. Valaee, J. Shang, and R. Zhang, "Locomotion activity recognition using stacked denoising autoencoders," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 2085–2093, 2018.

[46] Y. Dong, X. Li, J. Dezert, M. O. Khyam, M. Noor-A-Rahim, and S. S. Ge, "Dezert-smarandache theory-based fusion for human activity recognition in body sensor networks," *IEEE Trans. Ind. Inform.*, vol. 16, no. 11, pp. 7138–7149, 2020.

[47] Y. Tang, Q. Teng, L. Zhang, F. Min, and J. He, "Layer-wise training convolutional neural networks with smaller filters for human activity recognition using wearable sensors," *ArXiv Preprint arXiv:2005.03948*, 2020.

[48] S. Xu, L. Zhang, W. Huang, H. Wu, and A. Song, "Deformable convolutional networks for multimodal human activity recognition using wearable sensors," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–14, 2022.

[49] S. Xia, L. Chu, L. Pei, Z. Zhang, W. Yu, and R. C. Qiu, "Learning disentangled representation for mixed- reality human activity recognition with a single imu sensor," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–14, 2021.

[50] W. Gao, L. Zhang, W. Huang, F. Min, J. He, and A. Song, "Deep neural networks for sensor-based human activity recognition using selective kernel convolution," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–13, 2021.

[51] S. Abbaspour, F. Fotouhi, A. Sedaghatbaf, H. Fotouhi, M. Vahabi, and M. Linden, "A comparative analysis of hybrid deep learning models for human activity recognition," *Sensors*, vol. 20, no. 19, pp. 1–14, 2020.

[52] O. M. Saad and Y. Chen, "Capsphase: Capsule neural network for seismic phase classification and picking," *IEEE Trans. Geosci. Remote*, vol. 60, pp. 1–11, 2022.

[53] X. Sun, H. Xu, Z. Dong, L. Shi, Q. Liu, J. Li, T. Li, S. Fan, and Y. Wang, "Capsganet: Deep neural network based on capsule and gru for human activity recognition," *IEEE Sys. J.*, vol. 16, no. 4, pp. 5845–5855, 2022.

[54] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proc IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit. (CVPR)*, 2020, pp. 9726–9735.

[55] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. Int. Conf. Machin. Learn.*, 2020, pp. 1575–1585.

[56] H. Zhao, X. Yang, C. Deng, and D. Tao, "Unsupervised structure-adaptive graph contrastive learning," *IEEE Trans. Neur. Net. Lear.*, pp. 1–13, 2023.

[57] D. Feng, S. Han, H. Xu, X. Liang, and X. Tan, "Point-guided contrastive learning for monocular 3-d object detection," *IEEE Trans. Cybernetics*, vol. 53, no. 2, pp. 954–966, 2023.

[58] X. He, L. Fang, M. Tan, and X. Chen, "Intra- and inter-slice contrastive learning for point supervised oct fluid segmentation," *IEEE Trans. Image Process.*, vol. 31, pp. 1870–1881, 2022.

[59] Z. Wang, Q. Liu, and Q. Dou, "Contrastive cross-site learning with redesigned net for covid-19 ct classification," *IEEE J. Biomed. Health*, vol. 24, no. 10, pp. 2806–2813, 2020.

[60] H. Jung, Y. Oh, S. Jeong, C. Lee, and T. Jeon, "Contrastive self-supervised learning with smoothed representation for remote sensing," *IEEE Geosci. Remote S.*, vol. 19, pp. 1–5, 2022.

[61] Y. Su, X. Han, Y. Lin, Z. Zhang, Z. Liu, P. Li, J. Zhou, and M. Sun, "Css-lm: A contrastive framework for semi-supervised fine-tuning of pre-trained language models," *IEEE/ACM Trans. Audio Spe.*, vol. 29, pp. 2930–2941, 2021.

[62] Y. Liu, Z. Li, S. Pan, C. Gong, C. Zhou, and G. Karypis, "Anomaly detection on attributed networks via contrastive self-supervised learning," *IEEE Trans. Neur. Net. Lear.*, vol. 33, no. 6, pp. 2378–2392, 2022.

[63] J. Yu, H. Oh, M. Kim, and J. Kim, "Weakly supervised contrastive learning for unsupervised vehicle reidentification," *IEEE Trans. Neur. Net. Lear.*, pp. 1–11, 2023.

[64] G. Hinton, O. Vinyals, and J. Dean, "Distillation the knowledge in a neural network," *arXiv preprint arXiv: 1503.02531*, 2015.

[65] Z. Feng, J. Lai, and X. Xie, "Resolution-aware knowledge distillation for efficient inference," *IEEE Trans. Image Process.*, vol. 30, pp. 6985–6996, 2021.

[66] W.-C. Kao, H.-X. Xie, C.-Y. Lin, and W.-H. Cheng, "Specific expert learning: Enriching ensemble diversity via knowledge distillation," *IEEE Trans. Cybernetics*, vol. 53, no. 4, pp. 2494–2505, 2023.

[67] H. Zhao, X. Sun, J. Dong, C. Chen, and Z. Dong, "Highlight every step: Knowledge distillation via collaborative teaching," *IEEE Trans. Cybernetics*, vol. 52, no. 4, pp. 2070–2081, 2022.

[68] A. Romero, N. Ballas, S. Kahou, A. Chassang, C. Gatta, and Y. Bengio, “Fitnet: hints for thin deep nets,” in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–13.

[69] Y. Zhang, Z. Yan, X. Sun, X. Lu, J. Li, Y. Mao, and L. Wang, “Bridging the gap between cumbersome and light detectors via layer-calibration and task-disentangle distillation in remote sensing imagery,” *IEEE Trans. Geosci. Remote*, vol. 61, pp. 1–18, 2023.

[70] Z. Hao, Y. Luo, Z. Wang, H. Hu, and J. An, “Cdfkd-mfs: Collaborative data-free knowledge distillation via multi-level feature sharing,” *IEEE Trans. Multimedia*, vol. 24, pp. 4262–4274, 2022.

[71] L. Yu, V. Yazici, X. Liu, J. Weijer, Y. Chen, and A. Ramisa, “Learning metrics from teachers: compact networks for image embedding,” in *Proc IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit. (CVPR)*, 2019, pp. 2902–2911.

[72] J. Gou, L. Sun, B. Yu, S. Wan, W. Ou, and Z. Yi, “Multilevel attention-based sample correlations for knowledge distillation,” *IEEE Trans. Ind. Inform.*, vol. 19, no. 5, pp. 7099–7109, 2023.

[73] A. Yao and D. Sun, “Knowledge transfer via dense cross-layer mutual-distillation,” in *Proc. Lect. Notes Comput. Sci., ECCV*, 2020, pp. 294–311.

[74] J. Liu, X. Liu, H. Lin, B. Xu, Y. Ren, Y. Diao, and L. Yang, “Transformer-based capsule network for stock movements prediction,” in *Proc. FinNLP@IJCAI 2019*, 2019, pp. 66–73.

[75] T. Saha, S. R. Jayashree, S. Saha, and P. Bhattacharyya, “Bert-caps: A transformer-based capsule network for tweet act classification,” *IEEE Trans. Comput. Social Sys.*, vol. 7, no. 5, pp. 1168–1179, 2020.

[76] S. Duan, J. Cao, and H. Zhao, “Capsule-transformer for neural machine translation,” *arXiv preprint arXiv: 2004.14649*, 2020.

[77] T. DeVries and G. W. Taylor, “Improved regularization of convolutional neural networks with cutout,” *arXiv preprint arXiv:1708.04552*, 2017.

[78] Q. Li, B. He, and D. Song, “Model-contrastive federated learning,” in *Proc IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit. (CVPR)*, 2021, pp. 10 708–10 717.

[79] J.-L. Reyes-Ortiz, L. Oneto, A. Sam, X. Parra, and D. Anguita, “Transition-aware human activity recognition using smartphones,” *Neurocomputing*, vol. 171, pp. 754–767, 2016.

[80] J. R. Kwapisz, G. M. Weiss, and S. Moore, “Activity recognition using cell phone accelerometers,” in *Proc. 14th International Workshop on Knowledge Discovery from Sensor Data (at KDD-10)*, 2010, pp. 72–84.

[81] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, “Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine,” in *Proc. Lect. Notes Comput. Sci., IWAAL*, 2012, pp. 216–223.

[82] M. Gonźalez, C. Bergmeir, I. Triguero, Y. Rodíguerz, and J. Benítez, “Self-labeling techniques for semi-supervised time series classification: an empirical study,” *Knowl. Inf. Syst.*, vol. 55, pp. 493–528, 2018.

[83] H. Wang, Q. Zhang, J. Wu, S. Pan, and Y. Chen, “Time series feature learning with labeled and unlabeled data,” *Pattern Recogn.*, vol. 89, pp. 55–66, 2019.

[84] Y. Chen, B. Hu, E. Keogh, and G. E. Batista, “Dtw-d: Time series semi-supervised learning from a single example,” in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2013, pp. 383–391.

[85] Y. Zhou, Q. She, Y. Ma, W. Kong, and Y. Yang, “Transfer of semi-supervised broad learning system in electroencephalography signal classification,” *Neural Computing and Applications*, vol. 33, pp. 10 597–10 613, 2021.

[86] Q. Xie, Z. Dai, E. Hovy, M.-T. Luong, and Q. V. Le, “Unsupervised data augmentation for consistency training,” in *Proc. Adv. Neural Inf. Proces. Syst.*, 2020, pp. 1–13.

[87] S. Bhattacharya, P. Nurmi, N. Hammerla, and T. Plötz, “Using unlabeled data in a sparse-coding framework for human activity recognition,” *Pervasive Mob. Comput.*, vol. 5, pp. 242–262, 2014.

Zhiwen Xiao (M) received the B.Eng. degree in network engineering from the Chengdu University of Information Technology, Chengdu, China, and the M.Eng. degree in computer science from the Northwest A & F University, Yangling, China. He is pursuing the Ph.D. degree in computer science at Southwest Jiaotong University, Chengdu, China. His research interests include semantic communication, federated learning (FL), representation learning, data mining, and computer vision.

Huagang Tong received his Ph.D. degree in management sciences from Nanjing University of Aeronautics and Astronautics. He is with College of Economic and Management, Nanjing Tech University, Puzhu Road(S), Nanjing 211816, China. His research interests are knowledge discovery, operation research, and data mining.

Rong Qu (SM’12) is a full Professor at the School of Computer Science, University of Nottingham. She received her Ph.D. in Computer Science from The University of Nottingham, U.K. in 2003. Dr. Qu is an associated editor at Engineering Applications of Artificial Intelligence, IEEE Computational Intelligence Magazine, IEEE Transactions on Evolutionary Computation, Journal of Operational Research Society and PeerJ Computer Science. She is the Vice-Chair of Evolutionary Computation Task Committee since 2019 and Technical Committee on Intelligent Systems Applications at IEEE Computational Intelligence Society. She has guest edited special issues on the automated design of search algorithms and machine learning at the IEEE Transactions on Pattern Analysis and Machine Intelligence and IEEE Computational Intelligence Magazine. Her research interests include the modelling and optimisation, personnel scheduling, network routing, machine learning, and data mining.

Huanlai Xing (M) received Ph.D. degree in computer science from University of Nottingham (Supervisor: Dr Rong Qu), Nottingham, U.K., in 2013. He was a Visiting Scholar in Computer Science, The University of Rhode Island (Supervisor: Dr. Haibo He), USA, in 2020-2021. Huanlai Xing is with the School of Computing and Artificial Intelligence, Southwest Jiaotong University (SWJTU), and Tangshan Institute of SWJTU. He was on Editorial Board of SCIENCE CHINA INFORMATION SCIENCES. He was a member of several international conference program and senior program committees, such as ECML-PKDD, MobiMedia, ISCIT, ICCC, TrustCom, IJCNN, and ICSINC. His research interests include semantic communication, representation learning, data mining, reinforcement learning, machine learning, network function virtualization, and software defined networking.

Shouxi Luo (M) received the bachelor's degree in communication engineering and the Ph.D. degree in communication and information systems from the University of Electronic Science and Technology of China, Chengdu, China, in 2011 and 2016, respectively. He is an Associate Professor with the School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu. His research interests include data center networks, software-defined networking, and networked systems.

Zonghai Zhu received his B.Sc. degree in the Department of Information, Mechanical and Electrical Engineer, Shanghai Normal University, China, in 2010, and received his Ph.D. degree from the Department of Computer Science and Engineering, East China University of Science and Technology, China, in 2021. He is now an Assistant Professor in the School of Computing and Artificial Intelligence, Southwest Jiaotong University, China. His research interests include imbalanced problems, kernel-based methods, and graph-structured data.

Fuhong Song received the M.Eng. degree in computer technology and the Ph.D. degree in computer science and technology from Southwest Jiaotong University, Chengdu, China, in 2018 and 2022, respectively. He is currently a Lecturer with the School of Information, Guizhou University of Finance and Economics. His research interests include edge computing, multi-objective optimization and reinforcement learning.

Li Feng received his PhD degree from Xi'an Jiaotong University under the supervision of Prof. Xiaohong Guan (Academian of CAS, IEEE Fellow). He is a Research Professor and PhD supervisor with the School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu. His research interests include artificial intelligence, cyber security and its applications.