Similarity Embedding Networks for Robust Human Activity Recognition

CHENGLIN LI, CARRIE LU TONG, DI NIU, and BEI JIANG, University of Alberta
XIAO ZUO, LEI CHENG, JIAN XIONG, and JIANMING YANG, Tencent

Deep learning models for human activity recognition (HAR) based on sensor data have been heavily studied recently. However, the generalization ability of deep models on complex real-world HAR data is limited by the availability of high-quality labeled activity data, which are hard to obtain. In this article, we design a similarity embedding neural network that maps input sensor signals onto real vectors through carefully designed convolutional and Long Short-Term Memory (LSTM) layers. The embedding network is trained with a pairwise similarity loss, encouraging the clustering of samples from the same class in the embedded real space, and can be effectively trained on a small dataset and even on a noisy dataset with mislabeled samples. Based on the learned embeddings, we further propose both nonparametric and parametric approaches for activity recognition. Extensive evaluation based on two public datasets has shown that the proposed similarity embedding network significantly outperforms state-of-the-art deep models on HAR classification tasks, is robust to mislabeled samples in the training set, and can also be used to effectively denoise a noisy dataset.

CCS Concepts: • **Computing methodologies** → **Machine learning**; **Neural networks**; *Activity recognition and understanding*;

Additional Key Words and Phrases: Human activity recognition, embedding network, pairwise loss, noise robust

ACM Reference format:
Chenglin Li, Carrie lu Tong, Di Niu, Bei Jiang, Xiao Zuo, Lei Cheng, Jian Xiong, Jianming Yang, 2021. Similarity Embedding Networks for Robust Human Activity Recognition. *ACM Trans. Knowl. Discov. Data* 15, 6, Article 98 (May 2021), 17 pages.
https://doi.org/10.1145/3448021

1 INTRODUCTION

Human activity recognition (**HAR**) has become a fundamental capability in a wide range of **Internet of Things** (**IoT**) applications, including human health and well being [1], mobile security [25, 38], tracking and imaging [21], and vehicular road sensing [12, 15]. The widespread deployment of sensor technology, especially motion sensors like the accelerometer and gyroscope on mobile phones and smart watches, has made mobile sensing ubiquitous and fueled HAR research with data.

Authors' addresses: C. Li, C. L. Tong, D. Niu, and B. Jiang, University of Alberta, Edmonton, AB T6G 2R3, Canada; emails: {ch11, dniu}@ualberta.ca; X. Zuo, L. Cheng, J. Xiong, and J. Yang, Tencent, Shenzhen, Shennan Road 10000, 518057, China; emails: {royzuo, raycheng, janexiong, kimmyyang}@tencent.com.

1556-4681/2021/05-ART98 $15.00
https://doi.org/10.1145/3448021

HAR is essentially a time-series classification task based on a collection of heterogeneous measurements from multiple motion sensors. Traditional machine learning approaches such as decision trees, support vector machines [2], hidden Markov models [29], and random forest [26] models have been developed based on manually extracted features and achieved decent performance in some controlled environments [7]. However, these conventional pattern recognition algorithms heavily rely on the features correctly engineered from the raw data based on domain expertise and signal processing techniques, and therefore have limited generalizability.

Recently, deep learning has achieved great advances in fields such as computer vision [17] and natural language processing [23, 24]. Popular deep learning models, including **convolutional neural networks** (**CNNs**), **recurrent neural networks** (**RNNs**), **Restricted Boltzmann Machine** (**RBM**), and their hybrids, have also been adopted for HAR tasks [5, 27] to overcome the limitations of manual feature engineering. For example, DeepSense [41], as a deep model, has achieved the state-of-the-art HAR performance.

These studies have fine-tuned the architecture of the deep neural networks and achieved good prediction accuracies in carefully controlled environments. For example, the **University of Southern California Human activity Dataset (USC-HAD)** dataset [43] was collected from a special device, MotionNode, with a fixed sensing position. (All users collect data by wearing MotionNode at the same position.) Thus, even using CNNs with only four hidden layers, we can achieve a test accuracy of 97.01% [14]. On the other hand, the **HHAR** (**Heterogeneous Human Activity Recognition**) dataset collected by [34] is a more complex one, with heterogeneous sampling rates, hardware devices, and mobile operating systems. On this dataset, even with data augmentation techniques such as adding random noise, interpolation, and resampling [42], the state-of-the-art model, DeepSense, which combines RNNs with multiple convolutional layers, can only achieve a 94% accuracy. Therefore, more robust models need to be developed to handle increasingly complex and nonuniform sensor data from diverse real-world devices.

Furthermore, to date, two important factors have hindered the widespread deployment of deep models in real-world HAR applications. *First*, it is commonly known that deep neural networks need to be trained on a large amount of samples in order to yield competent performance. However, it is hard to collect a large amount of labeled samples to train HAR classifiers. Most HAR datasets reported in the literature contain no more than a few or a dozens of users (with the largest study based on 48 users [39]) and no more than tens of thousands of processed training samples, a size not comparable to the use of deep models in other fields, which usually have datasets of millions of samples or more, e.g., ImageNet. *Second*, in real-world HAR applications, training data are usually collected from crowdsourcing and may be subject to labeling mistakes or noise due to careless or malicious users. Furthermore, unlike images, speeches, or text, once sensor signals are collected, it is extremely hard to authenticate the correctness of their activity labels. As a result, for HAR problems, usually either a high-quality yet small dataset or a larger dataset with label noise is available. This explains why existing deep neural networks, which are originally designed for large and clean datasets, do not yield close-to-perfect performance on HAR tasks.

To overcome the aforementioned issues, in this article, we propose a robust ***similarity embedding network*** (**SEN**) that can generate discriminative vectorized embeddings of input signals based on a small amount of training data. Such embeddings can directly be used for HAR, greatly boosting the classification accuracy, or for data denoising. We have made multiple contributions:

First, similar to DeepSense, we leverage both CNNs and RNNs to model the interaction among different sensor signals within a same time interval and the sequential dependencies of signals across time intervals. Yet, we represent input signals differently and apply convolutional and **Long Short-Term Memory** (**LSTM**) layers in a new way that is optimized for performance. We show

that our network structure alone when directly used for activity recognition is comparable to other state-of-the-art deep models, even at a much lower sampling rate of sensor data.

More importantly, we introduce the SEN, which is based on the proposed neural network, yet trained by approximating the class label similarity between any two samples using the cosine similarity between their embeddings in a real space. This replaces the cross-entropy loss in the original classification problem by a pairwise loss between a pair of samples, such that the embeddings of samples from the same class are pushed to cluster, while those of different classes are pushed to separate from each other. Since every pair of samples yields a loss value to optimize the proposed SEN, we in fact have bootstrapped a small dataset into a larger amount of training samples, while building robustness into the model. We discuss the connections of our technique to as well as its differences from learning to hash [40], matching networks [36], and one-shot learning [16].

Based on the signal embeddings trained with the proposed SEN, we further propose two methods that achieve state-of-the-art activity recognition performance, including (1) a nonparametric nearest neighbor approach applied onto the embedded vectors; and (2) an MLP classifier applied on top of the SEN. We demonstrate, through experimental results, that the nonparametric nearest neighbor approach achieves strong robustness in the presence of small training sets and label noise.

In addition, we also propose a simple denoise procedure based on the trained SEN, which can leverage a small clean set to detect mislabeled samples in a potentially contaminated dataset.

We demonstrate the superiority and robustness of the proposed SEN on two publicly available datasets, the HHAR dataset and USC-HAD dataset. The proposed classification methods based on the embeddings found by SEN significantly outperform the state of the art in deep neural HAR. On the augmented HHAR dataset (following the same data augmentation procedure as DeepSense [42]), our proposed model architecture achieved an accuracy over 98%, outperforming the 94.2% accuracy reported by DeepSense in a leave-one-user-out evaluation scheme when trained with cross-entropy loss on an HHAR dataset. During stress test on the original HHAR dataset, by only using 110 training samples per class (7% of all data), our SEN-based method already achieves an accuracy of 95.03%, which is the same accuracy as achieved by traditional classification-oriented deep models, which, however, use 80% of data for training.

Furthermore, we demonstrate the robustness and generalizability of the proposed SENs by training the model on datasets with different noise rates. Results show that the proposed **SENs with similarity matching** (**SEN-SM**) method is much more robust to labeling noise compared to classification-oriented deep neural networks. Finally, we also demonstrate the usefulness of the proposed SEN in terms of denoising a heavily contaminated dataset (with 40% mislabeled samples) using embeddings trained on a small amount of clean data.

The remainder of this article is organized as follows. We first introduce the proposed network architecture, training based on pairwise losses, and our HAR classification methods in Section 2. Extensive evaluation of the proposed methods is then performed on two datasets and presented in Section 3. Section 4 discusses the related work. Finally, we conclude this article in Section 5.

2 SIMILARITY EMBEDDING NETWORK

In this section, we introduce the architecture of SEN, and the pairwise loss that we adopted to train the SEN model. Approaches that leverage the embedding output of SEN for an HAR task are also discussed in this section.

2.1 Sensor Signal Processing

In this article, an original data sample consists of readings from two motion sensors $S = \{S_1, S_2\}$, typically including the accelerometer S_1 and gyroscope S_2. Signals generated by each sensor S_i

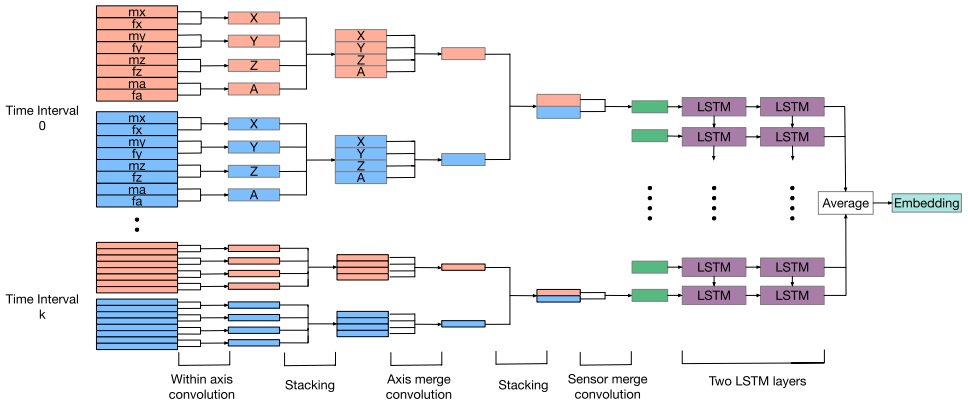


Fig. 1. The proposed embedding network architecture. Boxes in orange represent data from an accelerometer, while boxes in blue represent data from a gyroscope. mx, my, mz, and ma denote the magnitudes after Fourier transforms and fx, fy, fz, and fa denote the corresponding frequencies after Fourier transforms.

form a matrix D_i of the shape $d^i \times n^i$, where d^i is the dimensionality of the sensor S_i and n^i is the length of the signal. Typically, $d^1 = d^2 = 3$ for motion sensors. Our procedure to process input motion sensor signals is similar to yet different from other typical deep models like DeepSense. The differences will be explained at the end of this subsection.

Considering one particular sensor S_i as an example. Let x_i, y_i, and z_i be the time-series data of sensor S_i along the x, y, and z axes, respectively. We first compute the amplitude series as $a_i = \sqrt{x_i^2 + y_i^2 + z_i^2}$. The amplitude series a_i is then added to the original data of sensor S_i as its $(d^i + 1)$th axis. Thus, the data of sensor S_i become a matrix of shape $(d^i + 1) \times n^i$.

In order to leverage RNNs to model sequential dependencies, we split the data into k time intervals, each of width τ, which is a tunable hyper-parameter. The Fourier transform is then applied to the time series of each time interval in each of the $d^i + 1$ axes to get its frequency domain representation, which in our model is represented by both a vector of magnitudes and their corresponding frequencies. For example, the input series from axis x of sensor S_i is transformed into two vectors mx and fx, representing the magnitudes and their corresponding frequencies after the Fourier transform, respectively. Then, we stack all the outputs from Fourier transforms into a $2(d^i + 1) \times f$ matrix, where f is the length of frequency-domain representations. We apply the same procedure to all time intervals of sensor S_i and then stack them into a $k \times 2(d^i + 1) \times f$ tensor X^i. Finally, the set of resulting tensors for each sensor, $\mathcal{X} = \{X^i\}, i = 1, 2$, will be the input into our embedding network.

Note that our signal-processing method differs from typical deep models for HAR like DeepSense in two ways: (1) we introduce the amplitude series a_i; and (2) for each time interval at each sensor, we stack data into a $2(d^i + 1) \times f$ matrix with each of the $d^i + 1$ axes expanded into two rows (including magnitudes and frequencies after the Fourier transform), whereas DeepSense processes the same interval of data into a long $d^i \times 2f$ matrix using concatenation.

2.2 Network Architecture

Our embedding network encodes the input of any mobile sensor reading sample, which is a $k \times |S| \times 2(d + 1) \times f$ tensor $\mathcal{X}$ computed using the signal-processing procedure mentioned above, into an embedded vector. As is shown in Figure 1, the embedding network leverages both CNNs and RNNs in its neural encoding operation. Although prior research like DeepSense also adopted

CNNs and RNNs to encode the time series from sensors, our network structure distinguishes from previous studies in multiple aspects.

First, as has been mentioned above, the input of our network is a tensor of shape $k \times |S| \times 2(d + 1) \times f$. As is shown in the left side of Figure 1, in each interval, mx, my, mz, and ma represent the magnitudes (in frequency domain) and fx, fy, fz, and fa represent the corresponding frequencies. However, in DeepSense, the input data of each sensor form a tensor of shape $d \times 2f \times k$. That is, for each axis, DeepSense merges the output of the Fourier transform into a single vector, whereas we treat the magnitude–frequency pair as two stacked vectors. By stacking vertically, we can apply convolutional layers and stacking layers recursively and hierarchically for better feature extraction.

Second, similar to DeepSense, the data from each time interval are passed through a series of convolutional layers. But we apply the convolutional layers in a different way. DeepSense first applies three **one-dimensional (1D)** convolutional layers to the $d \times 2f$ input from each sensor to get a vector and, then, after stacking the vectors from different sensors, applies three other 1D convolutional layers to learn the interactions between sensors.

In contrast, we apply *hierarchical* convolutional schemes to the input of size $2(d + 1) \times f$ from each sensor. As is shown in Figure 1, first, we apply the *within-axis convolution* with a filter of size $2 \times conv1$ to the stacked pair of magnitudes and frequencies in each axis. Then, we stack data from the four axes x, y, z, and a together and apply the *axis-merge convolution* with a filter of size $4 \times conv2$ to learn the interactions among axes. Finally, we stack the produced vectors from different sensors and apply the *sensor merge convolution*, which consists of two convolutional layers with a $2 \times conv3$ filter and a $1 \times conv4$ filter, respectively. All the convolutions in our network are 1D convolutions, with filters scanning along one dimension horizontally. In contrast to DeepSense which has six convolutional layers, we only have four convolutional layers. By using hierarchical convolutions and within-axis convolutions, and by applying stacking repeatedly, we avoid running into a 1D vector too early.

After the data of all k time intervals have individually gone through the above hierarchical convolutional layers, we obtain k CNN outputs in the order of time intervals. An RNN is then applied to these CNN outputs to capture sequential dependencies. A stacked structure with two layers of **Gated recurrent unit (GRU)** is applied in DeepSense. Instead, we use two LSTM layers in our model, which leads to better performance according to experiments. Finally, the k outputs of the second LSTM are averaged to yield the final output of our network, which serves as the embedding of the input signal sample.

2.3 Training with Pairwise Losses

For the proposed SEN, key is to train it using pairwise losses instead of cross-entropy in classification. This will encourage the physical clustering of sample embeddings in a real space, bootstrap the training set, and make the model robust to noise. Given N training samples $X = \{x_i\}_{i=1}^N \in \mathbb{R}^{m \times N}$, where m is the input feature size, and their labels $Y = \{y_i\}_{i=1}^N \in \mathbb{R}^{c \times N}$, where y_i is a one-hot vector of length c, with c being the number of activities, our objective is to learn an embedding function $h(\cdot)$ represented by the neural network presented above so that the embeddings of all samples $\hat{X} = \{h(x_i)\}_{i=1}^N \in \mathbb{R}^{l \times N}$ are well clustered in the real space $\mathbb{R}^l$ according to their class labels.

Let $E = \{e_i\}_{i=1}^N$, where $e_i = h(x_i)$, represent the set of all the embedded vectors. Let s_{ij} denote the similarity between sample i and sample j, with $s_{ij} = 0$ if samples i and j are not from the same class; otherwise, $s_{ij} = 1$. With the label information in the training set, the pairwise similarity relationships $S = \{s_{ij}\}$ can be easily derived. The **Maximum a Posteriori (MAP)** estimation of

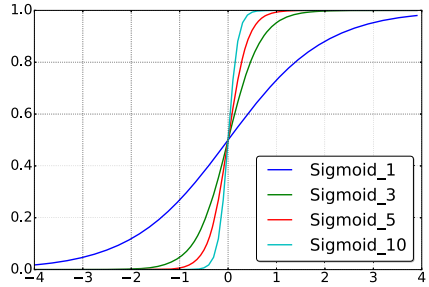


Fig. 2. Logistic sigmoid function with different k values.

the embeddings $E = e_i, i = 1, 2, \ldots, N$ can be represented as

$$p(E|S) \propto p(S|E)p(E) = \prod_{s_{ij} \in S} p(s_{ij}|e_i, e_j)p(e_i, e_j), \tag{1}$$

where $p(S|E)$ denotes the likelihood function. For each pair of the input data, $p(s_{ij}|E)$ is the conditional probability of s_{ij} given the embeddings E, which is defined as

$$p(s_{ij}|E) = \begin{cases} \sigma(\Phi_{ij}), & s_{ij} = 1 \\ 1-\sigma(\Phi_{ij}), & s_{ij} = 0 \end{cases}, \tag{2}$$

where $\sigma(x) = 1/(1 + e^{-kx})$ is the logistic sigmoid function with a tunable parameter k, and Φ_{ij} is the cosine similarity between embedding e_i and e_j defined as

$$\Phi_{ij} = \frac{e_i^T e_j}{|e_i| \cdot |e_j|}. \tag{3}$$

Obviously, if the conditional probability $p(s_{ij}|e_i, e_j)$ is maximized, the cosine similarity Φ_{ij} between the embedded vectors e_i and e_j will reach the maximum if they are from the same class and Φ_{ij} will reach the minimum if they are from different classes. It is worth noting that the value of the cosine similarity falls in the range of $[-1, 1]$, which is too narrow a section for the standard logistic sigmoid function where $k = 1$. To better distinguish capabilities of embedding pairs with different cosine similarities, in our model, we set k to be 10 so that we can cover almost the whole probability range from 0 to 1. As illustrated in Figure 2, when $k = 1$ the covered probability range is approximately from 0.3 to 0.7, while it is almost 0 to 1.0 when $k = 10$.

To maximize $p(S|E)$, the negative log-likelihood function J defined in (4) is used to learn the embeddings. Minimizing J by choosing the optimal embeddings networks that generate $E = e_i, i = 1, 2, \ldots, N$ will make the cosine similarity of any two samples from the same class as large as possible while in the mean time pushing the similarities between samples of different classes to as small as possible.

$$J = -\sum_{s_{ij} \in S} log\ p(s_{ij}|e_i, e_j) = -\sum_{s_{ij} \in S} (s_{ij} k \Phi_{ij} - log\ (1 + e^{k\Phi_{ij}})). \tag{4}$$

To minimize J, we can sample a batch of B pairwise samples, each in the form of (x_i, x_j, s_{ij}), to perform batched learning. For each training sample, (x_i, x_j, s_{ij}), $e_i = h(x_i)$ and $e_j = h(x_j)$ are obtained from the same embedding network $h(\cdot)$. And the parameters of $h(\cdot)$ will be updated by error back-propagation according to (4). Popular optimization algorithms such as stochastic gradient descent (SGD), Adam, and AdaGrad can be adopted to minimize J.

There are many studies in the field of learning to hash [20, 40] (mostly applicable to images) that also minimize pairwise losses. However, they are different from our method in the following aspects. First, studies in learning to hash focus on similarity preserving from the original feature

space to the hash space. The similarity in the original space is often measured by l_2-norm or cosine distance of raw features. While in our model, the original similarity to be preserved is whether two samples belong to the same class. Second, the embeddings in learning to hash are binary vectors, where the hamming distance is used to measure the similarity between two embedded vectors, mainly to speed up retrieval. On the other hand, the embeddings from our network are real vectors such that the cosine similarity can be used to maximize the separation of across-class samples. Third, the problem settings of learning to hash are also different: they aim to speed up information retrieval, especially of image objects, and do not suffer from a lack of training data or noisy data with mislabeled samples.

Our work is also related to matching networks [36] and one-shot learning [16], which perform image recognition based on a small number of samples. Matching networks [36] still use cross-entropy loss between predicted labels and the ground truth labels to train the network, while we use the pairwise similarities between samples to learn embeddings. The prediction of a new sample relies on the selected support set. One-shot learning uses pairwise losses in a Siamese network. However, it minimizes a cross-entropy loss and uses the $l1$-distance measure of almost binary embeddings (similar to hamming distance), while we maximize the log likelihood of observing the similarities given embeddings and embed vectors in a real space to maximize separability.

2.4 Activity Recognition

To leverage the proposed SENs for an HAR task, we propose two algorithms, which take the embeddings of samples calculated by a pretrained SEN as inputs. By minimizing the pairwise loss defined in (2.3), the embeddings of all the samples are expected to be well clustered. Hence, the most straightforward way to predict the label of a newly arrived sample is through the nonparametric ***k*-nearest neighbor (k-NN)** algorithm. To speed up prediction and also enhance robustness to noise, we propose to perform similarity matching between the new sample and the class centers in the embedded space. That is, the label for a new sample is predicted to be the label of the class center that is the closest to the new sample in the embedded space. The class center is calculated by averaging the normalized embeddings of all training samples in that class, with the distance measure being cosine distance. If the label noise is randomly scattered in the embedded space, its impact to each class center is reduced. Thus, we argue that this SEN-SM method for HAR is robust to label noise and can achieve high accuracy even if it is trained on a noise dataset. We will further show experimental results in the following section to show the robustness of the SEN-SM method.

In addition, we also propose to use another **multilayer perceptron (MLP)** model with one hidden layer, to predict a sample's label only based on its embedding vector without using training data in prediction. We call this HAR method SEN-MLP. Here the MLP layer is trained *separately* from the embedding network on the same training dataset. With the final MLP layer, the whole model structure looks similar to a typical deep model for classification. However, they are essentially different. In our model, the embedding network and the MLP layer are trained separately, with sampled pairwise losses and the cross-entropy loss, respectively, whereas in typical classification deep models, they are jointly trained only with the cross-entropy loss, which does not emphasize the separability or clustering of the embeddings. As the MLP layer is trained separately with the same training data, the SEN-MLP algorithm does not have noise robustness.

The SEN-MLP method have both advantages and disadvantages over the SEN-SM approaches. To begin with, as a nonparametric method, SEN-SM is simple, fast, and robust to label noise training samples. However, the calculation of class centers is hard when the dataset is huge, causing the updating of class centers time-consuming; thus, the SEN-SM model is not suitable for on-line learning applications where model updating is common. For the SEN-MLP method, even though the last MLP layer need to be trained separately from the SEN model and it is not noise robust, it

Table 1. Details of the Original and Processed Public Datasets

Dataset	#Activity	#Users	S.Rate	Position	#Samples
USC-HAD	12	14	100 Hz	Hip	–
HHAR	6	9	50–250 Hz	Waist	–
Processed data	#Activity	#Users	S.Rate	Position	#Samples
USC-HAD	6	14	25 Hz	Hip	3,170
HHAR	6	9	25 Hz	Waist	9,335

is much convenient to update the weights of the last MLP layer compared to the updating of class centers in the on-line learning settings.

As mentioned above, if the MLP layer and SENs are trained *jointly* with the cross-entropy end-to-end manner, instead of separately training with pairwise loss and cross-entropy loss, respectively, we get the Baseline method. The last MLP layer or fully connected layer projects the embeddings to class labels as shown in (5), where e_i is the embedding of sample i from the SEN's part, y_i' is the predicted label, $\sigma(\cdot)$ is the softmax function, and $g(\cdot)$ is the activation function with b and b_0 being the bias:

$$y_i' = \sigma(b_0 + g(W \cdot e_i + b)). \tag{5}$$

This Baseline algorithm is similar to previous algorithms such as DeepSense and the DCNN model in [14]. These models are all trained with an end-to-end fashion by minimizing the averaged cross-entropy loss on the training dataset (for classification), shown in (6), where y_i is the true label of sample i, c is the number of classes, and N is the number of samples in the training set:

$$loss = \frac{1}{N} \sum_{i}^{N} \sum_{j}^{c} -y_{ij} log(y_{ij}'). \tag{6}$$

We will show the superiority of our baseline method with proposed neural network architecture compared to previous work such as DeepSense with leave-one-user-out results.

3 EXPERIMENTS

In this section, experimental results on two public datasets are presented to show the efficiency and robustness of the proposed SENs for human activity recognition.

3.1 Datasets and Experiment Setup

There are many publicly available datasets that are widely adopted by recent studies on HAR [11, 28, 39]. In this article, we evaluate our proposed and baseline methods on two public datasets reported by [43] and [34], respectively. The detailed information on these two datasets is provided in Table 1. The *USC-HAD* [43] dataset was collected in 2012 by MotionNode, a wired sensing platform, with a sampling rate of 100 Hz, while the *HHAR* [34] dataset was collected in 2015 with eight different types of smartphones with sampling rates varying from 50 to 200 Hz.

Preprocessing is applied to maintain consistency between the two datasets and reformat the original data to fit our proposed architecture. We first select six types of activities, including "Standing," "Sitting," "Walking," "Upstairs," "Downstairs," and "Running" from the USC-HAD dataset. While in the HHAR dataset, we have samples from activities "Standing," "Sitting," "Walking," "Upstairs," "Downstairs," and "Biking." We then down-sample all sensor readings in both datasets to 25 Hz, which is a feasible sampling rate on everyday mobile devices with lower overhead. After down-sampling, the long consecutive sensor data (up to 5 minutes) are segmented into 6-second sensor readings. Thus, the HAR task in our experiments is to recognize the activity type for each

given 6-second sample. We have obtained 9,335 and 3,170 6-second samples from the HHAR and USC-HAD datasets, respectively. Finally, we split each of the two datasets into two parts, including the training (80%) and testing (20%) sets. All models are trained on the entire (or a portion of the) training set and evaluated on the corresponding test set. However, in DeepSense on the HHAR dataset, apart from sensor readings generated by a smartphone, signals from smartwatches are also included in their evaluation process. They also augmented the original dataset by adding noise, interpolation, and resampling, to get a dataset large enough for training deep neural networks. Therefore, to have a fair comparison to results reported by DeepSense, we also evaluate our classification-oriented model on the augmented dataset.

Experiments with several approaches are conducted to demonstrate the benefits of the proposed SEN, including the following algorithms:

- **SEN-SM:** similarity matching on top of the proposed SEN.
- **SEN-MLP:** an MLP with one hidden layer, trained separately, applied on top of the proposed SEN.
- **DeepSense:** the DeepSense model proposed by Yao et al. [41].
- **Baseline:** the same network structure as SEN-MLP, yet is trained jointly end-to-end with cross-entropy based on the label prediction error. This is similar to DeepSense in terms of end-to-end cross-entropy training, although differing from DeepSense in detailed network architecture.

As we have claimed that the proposed SEN can be efficiently trained even with a small number of training samples, we further conduct stress tests to explore the ability of SEN under small training sets. To do this, we randomly choose a small number of samples for each class from the entire training set, which is 80% of the original dataset. We then train the SEN and test its HAR performance on the test set with the proposed SEN-SM method. For stress tests, the number of training samples selected varies from 30 to 200.

Furthermore, to evaluate the robustness of the proposed SEN-SM algorithm to training samples with noisy labels, we perturb the training samples by randomly changing their labels and showing the generalization ability of SEN-SM on a noise-free test set. On the USC-HAD dataset, different noise rates, with 10%–40% of the samples perturbed, are adopted to evaluate the robustness of the proposed model. However, on the HHAR dataset, to avoid showing repeated results, we only evaluate the model when it is trained under the dataset with 40% of the samples perturbed.

The accuracy, F_1 score, and averaged F_1 score are widely used in HAR research [6, 30] for performance evaluation. The averaged F_1 score is defined as

$$Avg.F_1 = \frac{\sum_{i=1}^{c} N_i \cdot F_1^i}{\sum_{i=1}^{c} N_i}, \tag{7}$$

where c is the number of classes, F_1^i is the F_1 score of the ith class on the test set and N_i is the number of samples in the ith class.

In addition, the SEN-SM can be further used for data denoise. We will discuss the denoise techniques and evaluation result in details in Section 3.5.

3.2 HAR Classification Results

To evaluate the efficiency of the proposed network architecture, following the same data processing procedure as DeepSense [42], we conducted the leave-one-user-out experiment on the augmented HHAR dataset with our Baseline method. Specifically, DeepSense augments the original dataset 10 times by adding Gaussian noise to original signals. On the augmented HHAR dataset, when trained with cross-entropy loss, our proposed network, SEN-MLP, can achieve an averaged

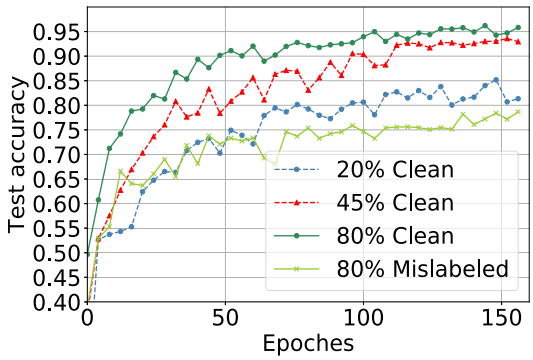
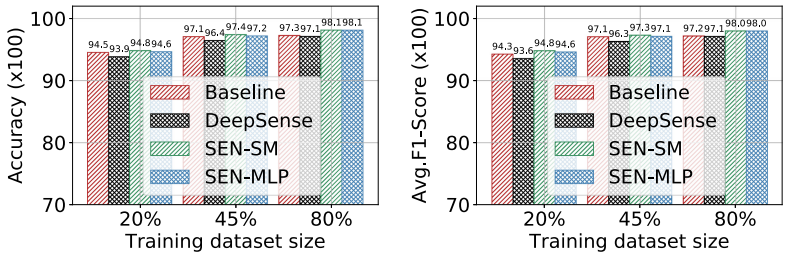


Fig. 3. Training curves of Baseline on different sizes of clean and noisy datasets from *HHAR*. 20%, 45%, and 80% denote the proportions of the entire HHAR dataset used. "Clean" and "Mislabeled" indicate whether the training set is clean or noisy. In the "Mislabeled" dataset, 40% samples are perturbed.



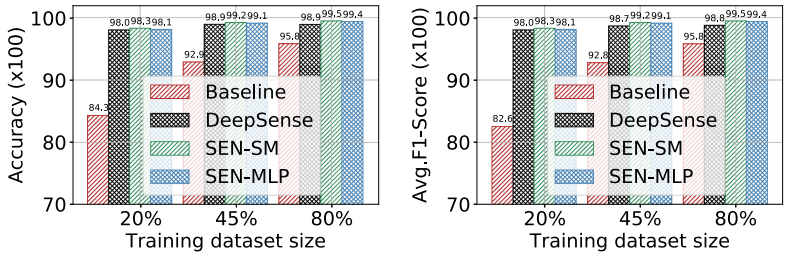
(a) Test accuracy on USC-HAD dataset (b) Averaged F_1 scores on USC-HAD dataset

Fig. 4. Test accuracy and averaged F_1 scores on the USC-HAD dataset of the proposed Baseline, SEN-MLP, and SEN-SM methods on different sizes of clean training sets. 20%, 45%, and 80% represent the proportions of the entire USC-HAD dataset used for training.

leave-one-out accuracy over 98%, which outperforms 94.2% reported by DeepSense with a great margin.

Then, to show the influence of the training dataset on the performance of a deep neural network, the Baseline method is evaluated on different sizes of training samples from the original HHAR dataset and even training samples with label noise. Test accuracy curves are shown in Figure 3, and the trend is obvious: as the size of the training set grows from 20% to 80%, the test accuracy improves from around 80% to near 96%. Furthermore, the model trained on the noise dataset gives the worst performance with an accuracy near 76% due to the fact that 40% of its samples are mislabeled.

Then, we compare the DeepSense and Baseline algorithms with our proposed SEN-MLP and SEN-SM methods. Detailed results on the USC-HAD and HHAR datasets are shown in Figure 4 and Figure 5, respectively. We can see as the training dataset size grows, both the accuracy and averaged F_1 score improve for all the three algorithms. However, SEN-SM and SEN-MLP always achieve superior performance. On the HHAR dataset, they achieved an accuracy of 98% and an averaged F_1 score above 0.98, even on the smallest dataset (20%), while the Baseline algorithm can only achieve an averaged F_1 score of 0.826 and an accuracy of 84.3%. When the training set size grows to 80%, the SEN-SM and SEN-MLP methods achieved an accuracy of 99% with an averaged F_1 score 0.99, while the Baseline method gives 96% accuracy and an averaged F_1 score 0.96. Similar results can also be found on the USC-HAD dataset, and SEN-SM and SEN-MLP still outperform the Baseline



(a) Test accuracy on HHAR dataset (b) Averaged F_1 scores on HHAR dataset

Fig. 5. The accuracy and averaged F_1 scores on the HHAR dataset of the proposed Baseline, SEN-MLP, and SEN-SM methods on different sizes of clean training sets. 20%, 45%, and 80% represent the proportions of the entire HHAR dataset used for training.

Table 2. Stress Test Results on the HHAR Dataset: the First Column Shows the Number of Training Samples for Each Class

# train data	Avg.F_1	Accuracy (%)	Precision(%)
30 (2%)	0.863	86.36	86.80
80 (5%)	0.926	92.62	92.62
110 (7%)	0.950	95.03	95.14
140 (9%)	0.964	96.36	96.38
170 (11%)	0.965	96.52	96.59
200 (13%)	0.978	97.81	97.82

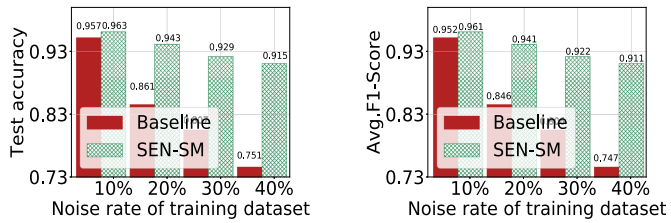
and DeepSense algorithms with an accuracy above 94% and an averaged F_1 score above 0.94 on 20% training data. However, on the USC-HAD dataset, one can see that the Baseline algorithm also achieved descent performance even on a small training dataset (20%), with an accuracy of 94.5% and an averaged F_1 score 0.943, which is due to the fact that the USC-HAD dataset is much simpler than the HHAR dataset with regard to both sensors used and sensing position.

Note that, on the USC-HAD dataset, our Baseline method achieved an accuracy of 97.3%, comparable to the **deep convolutional neural network** (**DCNN**) [14] trained on the original dataset with a much higher sampling frequency (100 Hz) with a reported accuracy of 97.1%, which is significant. Furthermore, on the HAR dataset, even though DeepSense is worse than our proposed SEN-SM and SEN-MLP algorithms, it still outperforms our Baseline method, which further demonstrates the superiority of the proposed SEN.

3.3 HAR Stress Testing on Small Training Sets

To further explore the limit of the proposed SEN, a stress test is conducted on the HHAR dataset, In the stress test, all models are trained on a limited number of training samples for each class and evaluated on the same test dataset. The number of training samples for each class ranges from 30 to 200.

The averaged F_1 score, accuracy, and precision scores are shown in Table 2. As the training dataset grows from 2% to 13%, the averaged F_1 score improved from 0.863 to 0.978. One can see when the number of samples for each class is only 110 (7%), we have already achieved an averaged F_1 score of 0.95, which is remarkable. Note that with only 30(2%) training samples (each being 6 seconds) available for each class, our method has already achieved an averaged F_1 score of 0.86, even better than what we get from the Baseline method with 20% of the training sample, which is



(a) Test accuracy (b) Averaged F_1 scores

Fig. 6. Test accuracy and averaged F_1 scores of proposed SEN-SM method and Baseline algorithm on the same training dataset with different noise rate. 10%, 20%, 30% and 40% represent the corresponding number of noise samples as their proportions in the whole dataset. Here the training samples are from 80% of the USC-HAD dataset.

0.826, as shown in Figure 5. This further shows the superiority of training embedding networks with pairwise loss compared to cross-entropy loss on a small training dataset.

3.4 Robustness to Label Noise

On the USC-HAD dataset, we trained both the SEN and Baseline models on a labeled noise dataset with different noise rates ranging from 10% to 40% to show the robustness and efficiency of our proposed method in the presence of label noise. The results are shown in Figure 6. It is obvious that at any noise level, the SEN-SM method achieved better performance than the Baseline model. Also, even though the noise rate is only 10%, the existence of label noise still leads to performance reduction for both the SEN-SM and the Baseline algorithms. As the noise rate grows, the performance becomes worse. When the noise rate grows from 10% to 40%, the test accuracy of the Baseline method declines dramatically from 95.7% to 75.1%, while that of the SEN-SM model reduces from 96.3% to 91.5%.

On the HHAR dataset, we evaluated both algorithms on a noisy dataset with a noise rate of 40% and found that the SEN-SM method achieved an averaged F_1 score of 0.84 with a 84.48% accuracy, while the Baseline method gave an averaged F_1 score of 0.73 and 77.25% accuracy. Compared to the original 99% and 96% accuracy achieved on a noise-free dataset, shown in Figure 5, our proposed model leads to less accuracy reduction and much better overall performance, which shows the robustness of our proposed SEN-SM method.

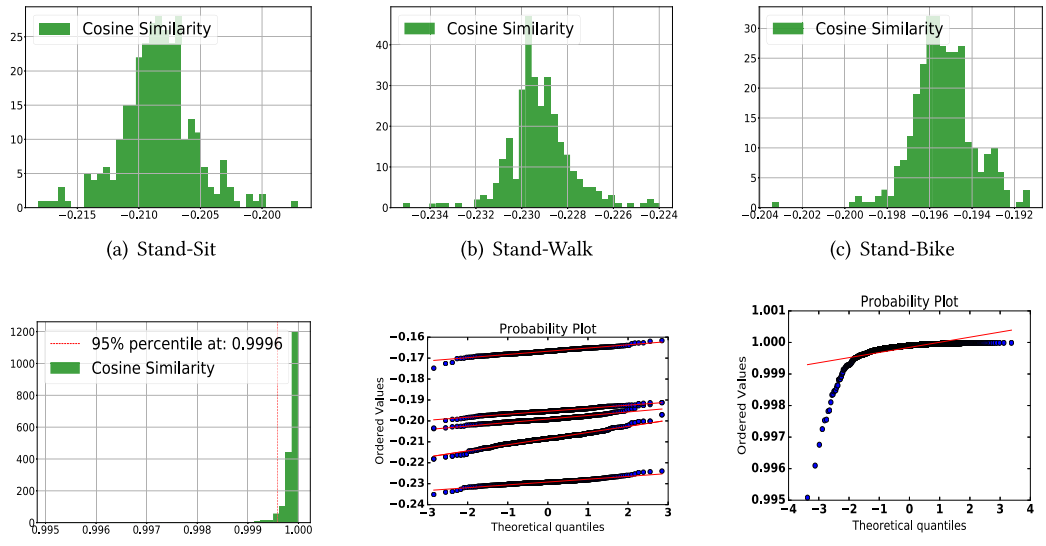
3.5 Data Denoising

Our proposed SEN can also be used to denoise a large noisy dataset, when trained on a small clean dataset. The goal of data denoising is to remove as many wrongly labeled samples as possible, so that the denoised data can be leveraged for training HAR models or other tasks.

We found that the between-class cosine distances (cosine similarities), i.e., cosine distances between each sample embedding and other class centers, follow a Gaussian distribution, while the in-class cosine distances, i.e., the cosine distance between each sample embedding and its own class center, highly concentrate around 1, as is suggested by Figures 7(a)–7(d). The Q–Q plots are provided to further show their distributions in Figures 7(e) and 7(f).

With this observation, we propose to filter out possibly mislabeled samples as follows. For any given sample, we consider it as correctly labeled if and only if

— its embedding is close enough to its own class center, i.e., its in-class cosine distance is no smaller than 5% of all the in-class cosine distances in that class;



(a) Stand-Sit (b) Stand-Walk (c) Stand-Bike

(d) In-Class (e) Cross-class Q-Q Plot (f) In-class Q-Q Plot

Fig. 7. The distribution and Q-Q plot of cosine similarity score. (a)-(c) show the distribution of cosine similarity between samples from "Stand" class and class centers of "Sit", "Walk" and "Bike". (d) shows the cosine similarity of all the samples and their corresponding class center. Q-Q plots of between class similarity scores are shown in (e). (f) shows the in-class cosine similarity Q-Q plot.

— its embedding is far from all the other class centers, i.e., its cosine distance to another class center is smaller than $\mu + 2\sigma$, where μ and σ are the mean and standard deviation of the corresponding between-class cosine distances.

In this manner, we can filter out noise samples with a low miss rate (high recall). Although some clean samples may also be filtered out, the idea of denoising is to detect as many mislabeled samples as possible to make sure the retrieved data are clean.

To evaluate denoising performance, we first train our SEN model on a randomly selected small dataset. Then, the in-class and between-class cosine similarity distance distributions are approximated on the same dataset. Finally, with the trained embedding neural network, we calculate class centers as well as the estimated statistics we can use to denoise a large, noisy dataset with a high recall score.

We evaluate this property by trying different numbers of clean samples selected from the HHAR dataset. Similar to stress tests mentioned above, we vary the number of samples for each class from 30 to 200 in the training set. Apart from the metrics mentioned above, recall score, which shows the detection rate of positive samples, is an important criterion when it comes to the data denoise problem, thus adopted for the evaluation of data denoise performance.

We evaluate the denoise performance with the manually perturbed labels as noise on the HHAR dataset. As shown in Table 3, the evaluation is also done on different sizes of clean training datasets like what we did in the stress test. As we can see, the larger the training set, the better the denoise performance. All the recall scores are close to 1 (almost no miss), while the denoise accuracy and F_1 score grow as the training dataset size grows from 30 to 200 samples per class. The F_1 scores and accuracy improved from 0.9 and 87% to 0.972 and 97% when the training dataset size grows from 30 to 200 samples per class, which also means that fewer clean samples will be mistakenly detected as noise.

Table 3. Denoising Results on the HHAR Dataset: the First Column Shows the Number of Training Samples for Each Class

# train data	Avg.F_1	Accuracy (%)	Recall (×100)
30 (2%)	0.90	88.01	99.05
80 (5%)	0.937	92.78	99.51
110 (7%)	0.957	95.17	99.80
140 (9%)	0.954	94.77	99.71
170 (11%)	0.972	96.86	99.80
200 (13%)	0.978	97.61	99.77

4 RELATED WORK

Some recent studies also apply deep neural network models to mobile sensing or HAR applications. [5] and [27] use a **Deep Boltzmann Machine** (**DBM**) and **Multimodal DBM** (**MultiDBM**) to improve the performance of heterogeneous human activity recognition. DeepEar [19] also uses a DBM to improve the performance of audio sensing tasks in an environment with background noise. IDNet [9] applies CNNs to the biometric gait analysis task. DeepX [18] and RedEye [22] reduce the energy consumption of deep neural networks, based on software and hardware, respectively. However, these studies do not capture the temporal relationship in time-series sensor inputs, and, with the only exception of the MultiRBM, all lack the capability of fusing multimodal sensor input. Zhu et al. [44] calculates the similarity degree between handcrafted recognition features and activity features to identify the most possible phone position, and then the result of this similarity matching is used for further activity recognition. Deepsense [41] applied RNN on top of CNN to acquire the sequential information of the input sensor data. Wan et al. [37] designs a smartphone inertial accelerometer-based architecture for HAR, which also uses CNN and RNN modules for better prediction accuracy. Those works all give out a classification model for HAR or other context awareness applications and cannot be used for denoise and have limited generalization performance when training data are insufficient or even noisy with wrong labels. Recently, Chen et al. [8] use a semi-supervised approach to deal with class imbalance in small labeled datasets. Bai et al. [4] proposed an unsupervised learning method, Motion2Vector, to convert a time interval of activity sensor data into a movement vector embedding in a multidimensional space. Instead of using an unsupervised or a semi-supervised training scheme, we adopt pairwise loss to train our proposed SEN, which can also alleviate the problem caused by a small amount of training samples.

Learning to hash has been attracting a large amount of research interest in machine learning and computer vision [40]. Applications of learning to hash include large-scale object retrieval [13], image classification [32], and detection [35]. Similarity preserving is the main methodology of learning to hash. Recent deep learning based studies in learning to hash have used deep neural networks to simultaneously learn the image representation and approximate the hashing function. **Convolutional Neural Network Hashing** (**CNNH**) [20] is one of the early works to incorporate deep neural networks into hash coding **Supervised Discrete Hashing** (**SDH**) [33], which directly optimizes the binary hash codes via the discrete cyclic coordinate descend method. The most relevant work is **Deep Supervised Discrete Hashing** (**DSDH**) [20], which uses CNN to learn the image representation and hash function simultaneously and output the binary hash encoding directly. However, DSDH uses hamming distance based pairwise loss and prediction loss as objectives, while we directly minimize the cosine distance based pairwise loss.

Some studies in metric learning that try to learn a transformation from input space to a low-dimension feature space [3] are also similar to our work. **Neighborhood Component Analysis**

(**NCA**) [10] learns a low-dimensional linear embedding of labeled data by directly maximizing a stochastic variant of the leave-one-out KNN. Salakhutdinov and Hinton [31] proposes methods to pretrain and fine-tune a multilayer neural network to learn a nonlinear transformation from input to feature space where nonparametric classification methods perform well. However, both the pretrain and fine-tune are done in an unsupervised manner, while in our network the training of the embedding network is supervised with pairwise similarity. Vinyals et al. [36] adopts a support set S for one-shot learning but is trained with the prediction loss of labels.

5 CONCLUSION

In this article, we propose a robust SEN to solve the main challenges faced when deploying complex deep models for HAR tasks in real world. Our SENs, even trained with cross-entropy loss on a dataset with much lower sampling rate, can already achieve a performance comparable to or even outperform most of the state-of-the-art algorithms for the HAR problem. By adopting pairwise loss, our model can generalize well even if trained on a small dataset or noisy data with mislabeled samples. Extensive experimental results on two publicly available datasets have demonstrated the superiority of our proposed embedding network model. Stress tests on a heterogeneous dataset shows the robustness of SEN to small training sets, while experiments on noisy datasets have shown its robustness to labeling noise in the training data. These capabilities are missing in the existing deep neural network models. Finally, even when trained on a small clean dataset, the proposed SEN is capable of denoising a heavily contaminated larger dataset with a 40% noise rate.

REFERENCES

[1] Hande Alemdar and Cem Ersoy. 2010. Wireless sensor networks for healthcare: A survey. *Computer Networks* 54, 15 (2010), 2688–2710.

[2] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge L. Reyes-Ortiz. 2012. Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In *Ambient Assisted Living and Home Care*, José Bravo, Ramón Hervás, and Marcela Rodríguez (Eds.). Springer, Berlin, 216–223.

[3] Christopher G. Atkeson, Andrew W. Moore, and Stefan Schaal. 1997. Locally weighted learning for control. In *Lazy Learning*. Springer, 75–113.

[4] Lu Bai, Chris Yeung, Christos Efstratiou, and Moyra Chikomo. 2019. Motion2Vector: Unsupervised learning in human activity recognition using wrist-sensing data. In *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers*. 537–542.

[5] Sourav Bhattacharya and Nicholas D. Lane. 2016. From smart to deep: Robust activity recognition on smartwatches using deep learning. In *Proceedings of the 2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*. IEEE, 1–6.

[6] Sourav Bhattacharya, Petteri Nurmi, Nils Hammerla, and Thomas Plötz. 2014. Using unlabeled data in a sparse-coding framework for human activity recognition. *Pervasive and Mobile Computing* 15 (2014), 242–262.

[7] Andreas Bulling, Ulf Blanke, and Bernt Schiele. 2014. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys* 46, 3 (2014), 33.

[8] Kaixuan Chen, Lina Yao, Dalin Zhang, Xianzhi Wang, Xiaojun Chang, and Feiping Nie. 2019. A semisupervised recurrent convolutional attention model for human activity recognition. *IEEE Transactions on Neural Networks and Learning Systems* 31, 5 (2019), 1747–1756.

[9] Matteo Gadaleta and Michele Rossi. 2016. Idnet: Smartphone-based gait recognition with convolutional neural networks. *Pattern Recognition* 74 (2018), 25–37.

[10] Jacob Goldberger, Geoffrey E. Hinton, Sam T. Roweis, and Ruslan R. Salakhutdinov. 2005. Neighbourhood components analysis. In *Proceedings of the 17th International Conference on Neural Information Processing Systems*. ACM, 513–520.

[11] Nils Y. Hammerla, Shane Halloran, and Thomas Ploetz. 2016. Deep, convolutional, and recurrent models for human activity recognition using wearables. In *IJCAI*.

[12] Shaohan Hu, Hengchang Liu, Lu Su, Hongyan Wang, Tarek F. Abdelzaher, Pan Hui, Wei Zheng, Zhiheng Xie, and John A. Stankovic. 2014. Towards automatic phone-to-phone communication for vehicular networking applications. In *Proceedings of the IEEE Conference on Computer Communications*. IEEE, 1752–1760.

[13] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. 2010. Aggregating local descriptors into a compact image representation. In *Proceedings of the 2010 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 3304–3311.

[14] Wenchao Jiang and Zhaozheng Yin. 2015. Human activity recognition using wearable sensors by deep convolutional neural networks. In *Proceedings of the 23rd ACM International Conference on Multimedia*. ACM, 1307–1310.

[15] Lei Kang, Bozhao Qi, Dan Janecek, and Suman Banerjee. 2015. EcoDrive: A mobile sensing and control system for fuel efficient driving. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*. ACM, 358–371.

[16] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *Proceedings of the 32 nd International Conference on Machine Learning*, Vol. 2.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60, 6 (2012), 84–90.

[18] Nicholas D. Lane, Sourav Bhattacharya, Petko Georgiev, Claudio Forlivesi, Lei Jiao, Lorena Qendro, and Fahim Kawsar. 2016. DeepX: A software accelerator for low-power deep learning inference on mobile devices. In *Proceedings of the 15th International Conference on Information Processing in Sensor Networks*. IEEE, 23.

[19] Nicholas D. Lane, Petko Georgiev, and Lorena Qendro. 2015. DeepEar: Robust smartphone audio sensing in unconstrained acoustic environments using deep learning. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 283–294.

[20] Qi Li, Zhenan Sun, Ran He, and Tieniu Tan. 2017. Deep supervised discrete hashing. In *Proceedings of the 31st Conference on Neural Information Processing Systems*. 2482–2491.

[21] Tianxing Li, Chuankai An, Zhao Tian, Andrew T. Campbell, and Xia Zhou. 2015. Human sensing using visible light communication. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*. ACM, 331–344.

[22] Robert LiKamWa, Yunhui Hou, Julian Gao, Mia Polansky, and Lin Zhong. 2016. RedEye: Analog ConvNet image sensor architecture for continuous mobile vision. In *Proceedings of the 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture*, Vol. 44. IEEE, 255–266.

[23] Bang Liu, Ting Zhang, Fred X. Han, Di Niu, Kunfeng Lai, and Yu Xu. 2018. Matching natural language sentences with hierarchical sentence factorization. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1237–1246.

[24] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association*.

[25] Emiliano Miluzzo, Alexander Varshavsky, Suhrid Balakrishnan, and Romit Roy Choudhury. 2012. Tapprints: Your finger taps have fingerprints. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*. ACM, 323–336.

[26] Tomáš Peterek, Marek Penhaker, Petr Gajdoš, and Pavel Dohnálek. 2014. Comparison of classification algorithms for physical activity recognition. In *Innovations in Bio-inspired Computing and Applications*, Ajith Abraham, Pavel Krömer, and Václav Snášel (Eds.). Springer International Publishing, Cham, 123–131.

[27] Valentin Radu, Nicholas D. Lane, Sourav Bhattacharya, Cecilia Mascolo, Mahesh K. Marina, and Fahim Kawsar. 2016. Towards multimodal deep learning for activity recognition on mobile devices. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*. ACM, 185–188.

[28] Daniele Ravi, Charence Wong, Benny Lo, and Guang-Zhong Yang. 2016. Deep learning for human activity recognition: A resource efficient implementation on low-power devices. In *Proceedings of the 2016 IEEE 13th International Conference on Wearable and Implantable Body Sensor Networks*. IEEE, 71–76.

[29] Charissa Ann Ronao and Sung Bae Cho. 2014. Human activity recognition using smartphone sensors with two-stage continuous hidden markov models. In *Proceedings of the 2014 10th International Conference on Natural Computation*. IEEE, 681–686. DOI: https://doi.org/10.1109/ICNC.2014.6975918

[30] Hesam Sagha, Sundara Tejaswi Digumarti, José del R. Millán, Ricardo Chavarriaga, Alberto Calatroni, Daniel Roggen, and Gerhard Tröster. 2011. Benchmarking classification techniques using the Opportunity human activity dataset. In *Proceedings of the 2011 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 36–40.

[31] Ruslan Salakhutdinov and Geoff Hinton. 2007. Learning a nonlinear embedding by preserving class neighbourhood structure. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics*. PMLR, 412–419.

[32] Jorge Sánchez and Florent Perronnin. 2011. High-dimensional signature compression for large-scale image classification. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 1665–1672.

[33] Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen. 2015. Supervised discrete hashing. In *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 37–45.

[34] Allan Stisen, Henrik Blunck, Sourav Bhattacharya, Thor Siiger Prentow, Mikkel Baun Kjærgaard, Anind Dey, Tobias Sonne, and Mads Møller Jensen. 2015. Smart devices are different: Assessing and MitigatingMobile sensing heterogeneities for activity recognition. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. ACM, 127–140.

[35] Andrea Vedaldi and Andrew Zisserman. 2012. Sparse kernel approximations for efficient classification and detection. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2320–2327.

[36] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra. 2016. Matching networks for one shot learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*. ACM, 3630–3638.

[37] Shaohua Wan, Lianyong Qi, Xiaolong Xu, Chao Tong, and Zonghua Gu. 2020. Deep learning models for real-time human activity recognition with smartphones. *Mobile Networks and Applications* 25, 2 (2020), 743–755.

[38] Chen Wang, Xiaonan Guo, Yan Wang, Yingying Chen, and Bo Liu. 2016. Friend or foe?: Your wearable devices reveal your personal pin. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*. ACM, 189–200.

[39] Jindong Wang, Yiqiang Chen, Shuji Hao, Xiaohui Peng, and Lisha Hu. 2018. Deep learning for sensor-based activity recognition: A survey. *Pattern Recognition Letters* 119 (2018), 3–11.

[40] Jingdong Wang, Ting Zhang, Nicu Sebe, Heng Tao Shen2018. A survey on learning to hash. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40, 4 (2018), 769–790.

[41] Shuochao Yao, Shaohan Hu, Yiran Zhao, Aston Zhang, and Tarek Abdelzaher. 2017. DeepSense: A unified deep learning framework for time-series mobile sensing data processing. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 351–360.

[42] yscacaca. 2019. *HHAR-Data-Process*. Retrieved from https://github.com/yscacaca/HHAR-Data-Process.

[43] Mi Zhang and Alexander A. Sawchuk. 2012. USC-HAD: A daily activity dataset for ubiquitous activity recognition using wearable sensors. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. ACM, 1036–1043.

[44] Yangda Zhu, Changhai Wang, Jianzhong Zhang, and Jingdong Xu. 2014. Human activity recognition based on similarity. In *Proceedings of the 2014 IEEE 17th International Conference on Computational Science and Engineering*. IEEE, 1382–1387.

Received May 2020; revised December 2020; accepted January 2021