Negative Selection by Clustering for Contrastive Learning in Human Activity Recognition

Jinqiang Wang®, Tao Zhu®, Member, IEEE, Liming Luke Chen®, Senior Member, IEEE, Huansheng Ning®, Senior Member, IEEE, and Yaping Wan

Abstract—Contrastive learning is an emerging and important self-supervised learning paradigm that has been successfully applied to sensor-based human activity recognition (HAR) because it can achieve competitive performance relative to supervised learning. Contrastive learning methods generally involve instance discrimination, which means that the instances are regarded as negatives of each other, and thus their representations are pulled away from each other during the training process. However, instance discrimination could cause overclustering, meaning that the representations of instances from the same class could be overly separated. To alleviate this overclustering phenomenon, we propose a new contrastive learning framework to select negatives by clustering in HAR, which is named Clustering for contrastive learning (ClusterCLHAR). First, ClusterCLHAR clusters the instance representations, and for each instance, only those from different clusters are regarded as negatives. Second, a new contrastive loss function is proposed to mask the same-cluster instances from the negative pairs. We evaluate ClusterCLHAR on three popular benchmark data sets: 1) USC-HAD; 2) MotionSense; and 3) UCI-HAR, using the mean F1-score as an evaluation metric for downstream tasks. The experimental results show that ClusterCLHAR outperforms all the state-of-the-art methods applied to HAR in self-supervised learning and semi-supervised learning.

Index Terms—Clustering, contrastive learning, human activity recognition, negative selection, sensor data.

I. INTRODUCTION

HUMAN activity recognition (HAR) based on the Internet of Things and wearable sensing (accelerometers and gyroscopes) has a crucial role in emerging user-centered smart applications, such as smart homes [1], [2], fall detection [3], [4], and healthcare rehabilitation [5], [6]. Recent applications of deep learning techniques [7], [8], [9], [10] have significantly improved activity recognition accuracy. However, deep learning methods usually require a large number of labeled data sets to train an activity recognition model. Manual labeling of sensor data is time consuming and tedious, especially in the healthcare field, where the collection of labeled data is more challenging. Moreover, the labels are affected by various noise sources, such as sensor noise, segmentation problems, and differences in the activities of different people, rendering the annotation process error-prone [11]. Therefore, insufficient data annotation becomes a major challenge for HAR.

To alleviate the issue of insufficient data annotation, contrastive learning, which is a paradigm of self-supervised learning, has been developed and indeed achieved excellent performance in computer vision [12]. The pretraining process for contrastive learning is to generate pseudolabels using data augmentation on a large amount of unlabeled data, enabling the model to distinguish which augmented instances are positive pairs and which augmented instances are negative pairs [13]. After pretraining, the model is fine-tuned in downstream tasks using a small amount of labeled data and can achieve performance comparable to supervised learning [14], [15].

There are many types of pretraining tasks for contrastive learning, such as MoCo [16], [17] and SimCLR [18], [19], which use instance discrimination [20] as the task. NNCLR [21], MSF [22], TTL [23], and HardCL [24] were proposed to redefine positive and negative pairs based on the instance discrimination task. Generally, the pretraining task of the aforementioned studies is to enable the model to generate close representations for positive pairs and distant representations for negative pairs in the latent space. However, such a pretraining task tends to force the model to overcluster instances of the same class into different clusters.

Pretraining tasks of contrastive learning applied to HAR also involve instance discrimination [20], [25], such as SimCLR-HAR [26] and CSSHAR [27], built on an improved SimCLR [18], and MoCoHAR [28] built on an improved MoCo [17]. These works generally optimize augmentation methods or backbone networks rather than pretraining tasks. Contrastive learning models that use instance discrimination as a pretraining task tend to fall into overclustering [23] during training. The normalized temperature-scaled cross-entropy loss (NT-Xent) is the contrastive loss function of SimCLR, which follows the instance discrimination task and easily
causes overclustering. The pretraining effect of NT-Xent is shown in Fig. 1(a). Each activity instance is separated under the latent space, even if they belong to the same class. This phenomenon occurs because the instance discrimination task requires pulling instance representations away from each other. As shown in Fig. 1(b), it is more ideal for HAR that the representations of activities of the same class are close to each other in the latent space. Such representations are more conducive to downstream classification tasks. Therefore, it is challenging to avoid overclustering during contrastive learning training. The challenge is how to avoid regarding the same-class samples as negative examples during the instance discrimination task.

To address this challenge, we extend SimCLR (Fig. 2(a) [18]) to propose a new contrastive learning framework named ClusterCLHAR to select negatives by clustering in HAR. ClusterCLHAR is shown in Fig. 2(b). First, ClusterCLHAR clusters the instance representations, and for each instance, only those from different clusters are regarded as negatives. Second, a new contrastive loss function Cluster-NT-Xent, which is built on an improved NT-Xent, is proposed to mask instances of the same cluster out of the negative pairs. Therefore, this function could reduce the probability of pulling away instance representations of the same class from each other. A graphical representation of the improvement of the contrastive loss function is shown in Fig. 3.

ClusterCLHAR is evaluated in the following experimental setup. Transformation prediction network (TPN) [29] is employed as the backbone network of ClusterCLHAR, which is fast in inference and outperforms DeepConvLSTM [30] in supervised learning. Three benchmark data sets, USC-HAD [31], MotionSense [32], and UCI-HAR [33], were employed for the experiment. First, we compare ClusterCLHAR with previous work under self-supervised learning (downstream tasks using all the data labels). The initial analysis shows that ClusterCLHAR outperforms all state-of-the-art self-supervised learning work on three data sets. Second, we evaluate ClusterCLHAR under semi-supervised learning (downstream tasks using a small fraction of the data labels). The pretrained model was fine-tuned using 1% or 10% of the data labels in the downstream task, and the remaining data were utilized as the test set. The semi-supervised experimental results show that ClusterCLHAR outperforms all state-of-the-art contrastive learning methods on three data sets. Last, we visualize the output of ClusterCLHAR and SimCLR using...
The remainder of this article is structured as follows. Section II reviews the work of contrastive learning on negative example selection and HAR. Section III describes the ClusterCLHAR framework in detail. Section IV designs self-supervised learning and semi-supervised learning experimental protocols to evaluate the performance of the proposed framework. Section V presents and discusses the experimental results of self-supervised learning and semi-supervised learning. Section VI further discusses the details of our proposed framework by ablation studies. Section VII summarizes this article and proposes future work based on the observed shortcomings.

II. RELATED WORKS

A. Selection of Negative Examples in Contrastive Learning

Contrastive learning is a paradigm of self-supervised learning. In contrastive learning, data augmentation is performed to generate pseudolabels that enable the model to distinguish between positive pairs and negative pairs in augmented samples [13]. Contrastive learning consists of three steps [14]. The first step is augmentation, which determines the quality of pseudolabel generation and has a significant impact on the final performance of the model. The second step is the encoder, which encodes the augmented samples and affects the quality of the generated representation in the latent space. The third step is the loss function (pretraining task), which defines the positive and negative samples that determine the direction the model will learn.

The definition of negative examples has a crucial role in the final performance of contrastive learning, and many studies on contrastive learning have been conducted on this issue. MoCo [16] uses queues to expand negative examples and achieves excellent performance with a small batch size. SimCLR [18] uses a larger batch size to expand the number of negative examples, which improves the performance of contrastive learning in a simple and efficient way. NNCLR [21] compares with SimCLR to select the most similar sample representation from a queue by the nearest neighbor method instead of the original representation to calculate the contrastive loss, thus improving the model performance by increasing the training complexity. MSF [22] calculates the contrastive loss by selecting the mean of the $K$ most similar sample representations from a queue instead of the original sample representation. TTL [23] and HardCL [24] define near positive examples as negative examples for the purpose of separating similar sample representations in the latent space.

The above work tends to cause overclustering, assigning the same class of samples to different clusters.

It is expected that the same-class sample representations remain close in the latent space while the different classes are pushed apart. For this reason, when designing the contrastive loss function, the possibility of pushing the same-class samples apart is reduced if we can avoid regarding the same-class samples as negative examples. The resulting encoder representation can approach the effect of Fig. 1(b), which will be easier and more efficient for the training of downstream classification tasks.

B. Contrastive Learning for HAR

Self-supervised learning has been applied to HAR based on sensor data, such as multitask SSL [29], CAE [36], Masked Reconstruction [37], and CPCHAR [38]. These studies use unlabeled data to generate pseudolabels and to set pretext tasks based on the pseudolabels, enabling the encoder to learn excellent representations by completing the pretext tasks. As a specific paradigm of self-supervised learning, contrastive learning has been applied to human activity recognition. SimCLR [26] first applies contrastive learning to HAR, using SimCLR’s architecture, and then achieves a slight improvement in activity recognition accuracy relative to supervised learning. CSSHR [27] replaces the backbone network with a custom transformer. Although the overall accuracy is improved, the performance improvement relative to the supervised learning of the backbone network is not significant. MoCoHAR [28] uses MoCo’s architecture and proposes...
a resampling sensor data augmentation method, which significantly improves the accuracy with a small amount of data labels compared with supervised learning. However, the contrastive loss function of the above work is InfoNCE [39] or NT-Xent [40]; both are essentially the instance discrimination task that will regard the same-class samples as negative examples. Such a pretraining task tends to cause overclustering so that the same class of activity representation is pulled apart.

To alleviate this issue, we improve the NT-Xent contrastive loss function by introducing an unsupervised clustering technique to mask the same-cluster samples. This approach avoids the same-cluster sample representation as negative examples to the extent that the same-cluster samples in negative examples are eliminated in calculating contrastive losses. This operation will enable the representation of the same-class activities to be closer in the latent space.

III. METHODS
A. Proposed ClusterCLHAR Framework

The framework of the proposed ClusterCLHAR is shown in Fig. 4. The steps in order are data augmentation, encoder, projection head, negative selection, and contrastive loss function. Our work focused on improving the last two components.

Negative Selection: As shown in the dashed box in Fig. 4, we optimize the negative selection method defined by the instance discrimination task. For the first branch, each sample representation was assigned a cluster label by using unsupervised clustering methods. Then representations with the same-cluster label will no longer constitute negative pairs. The similarity matrix of representations of the two branches is shown in the right part of Fig. 4. For each row of this matrix, the previous contrastive learning pretext task, such as SimCLR, is to maximize the similarity of positive pairs. Our proposed framework inherits this idea, but we do not define representations of the same-cluster as negative pairs. This change will potentially avoid the same-class sample representations being pulled apart under the latent space and effectively alleviate overclustering.

Contrastive Loss Function: Due to the change in the negative selection method, the corresponding contrastive loss function needs to be improved. The original contrastive loss function and the improvement process are detailed below.

The instance discrimination task of SimCLR, using NT-Xent [40] (normalized temperature-scaled cross-entropy loss, in 1) as the contrastive loss function, could cause overclustering and distance the representation of the same-class instances for the downstream classification task.

In (1), \((i,j)\) is a positive pair; \(z_i\) denotes the representation output by the projection head; \(N\) denotes the mini-batch length; \(I_{[\cdot]}\) is an indicator function, which is equal to 1 when the expression in \([\cdot]\) is true and 0 otherwise; and \(\tau\) is the temperature coefficient. The formula shows that sample \(i\) constitutes negative pairs for all samples except sample \(j\). This finding means that sample \(i\) will regard samples of the same class
as negative examples, which tends to cause overclustering of pretrained representations

\[
\ell_{i,j}^{\text{NT-Xent}} = - \log \frac{\exp(z_i \cdot z_j / \tau)}{\sum_{k=1}^{2N} I[k \neq i] \exp(z_i \cdot z_k / \tau)}.
\]  

(1)

To alleviate the phenomenon of overclustering and to avoid pulling away the same-class representations, a new contrastive loss function \(\text{Cluster-NT-Xent}\) is proposed, as shown in (2), where \(\text{cluster}(i)\) denotes the set of clusters in which the sample \(i\) is located. The batch calculation is shown in (3). We cluster the representations generated by the first branch and assign a cluster label to each representation. Representations with the same cluster label will no longer constitute negative pairs when computing the contrastive loss. The difference between \(\text{Cluster-NT-Xent}\) and \(\text{NT-Xent}\) is visually illustrated on the right part of Fig. 4. It can be seen that the sample representations with the same label no longer constitute negative pairs

\[
\ell_{i,j}^{\text{cluster}} = - \log \frac{\exp(z_i \cdot z_j / \tau)}{\sum_{k=1}^{2N} I[k \neq \text{cluster}(i)] \exp(z_i \cdot z_k / \tau)}.
\]  

(2)

\[
\mathcal{L}^{\text{Cluster}} = \frac{1}{2N} \sum_{k=1}^{N} \left[ \mathcal{L}_k^{\text{Cluster}}(2k-1,2k-2) + \mathcal{L}_k^{\text{Cluster}}(2k,2k-1) \right].
\]  

(3)

The ClusterCLHAR pretraining pseudocode is shown in Algorithm 1. After pretraining, the encoder is used for downstream tasks.

### B. Influence of the Hyperparameter \(\alpha\)

The performance of clustering methods has a significant impact on model performance. If the clustering results differ significantly from the true classification, then many true negative pairs will be eliminated, seriously affecting the pretraining performance. In clustering methods, samples close to the cluster centroids are more likely belong to the same class, than those close to the cluster boundaries [41]. Those samples at cluster boundaries are not suitable for our proposed negative selection strategy (samples of the same cluster do not constitute negative pairs) because they do not have a high probability of belonging to the same class. Therefore, we decided to reduce the proportion of samples using our proposed negative selection strategy to reduce the negative impact of the difference between the clustering results and the true classification. We set a hyperparameter \(\alpha\) indicating the proportion of samples close to the cluster center, in which the proportion of samples uses our proposed negative selection strategy and the rest of the samples use SimCLR’s strategy (instance discriminative). Fig. 5 shows the sample distribution using our strategy and SimCLR’s strategy at different \(\alpha\). Here, the samples with our proposed strategy use \(\text{Cluster-NT-Xent}\) as the loss function and the samples with SimCLR’s strategy use \(\text{NT-Xent}\) as the loss function

\[
\ell_{i,j}^{\text{Cluster-NT-Xent}} = - \log \frac{\exp(z_i \cdot z_j / \tau)}{\sum_{k=1}^{2N} I[k \neq \text{cluster}(i)] \exp(z_i \cdot z_k / \tau)}.
\]  

(4)

The contrastive loss function \(\text{Cluster-NT-Xent-}\alpha\) (normalized-temperature-scaled cross-entropy loss for selecting negatives by the hyperparameter \(\alpha\)) compatible with the above two strategies is shown in (4), where \(\text{thr}(\alpha, i)\) indicates that the value is 1 if sample \(i\) belongs to the top \(\alpha\)% of the sample representation closest to the cluster center and is 0 otherwise. Here, \(\alpha \in [0, 100]\) denotes the proportion of samples that are close to the cluster center, and these samples use our negative selection strategy. When \(\alpha\) decreases, the number of samples using our strategy decreases, and the number of negative pairs increases. Equation (4) not only sets the samples that do not belong to the cluster in which sample \(i\) is located as negative examples but also considers samples that belong to the cluster in which sample \(i\) is located but do not belong to the top \(\alpha\)% near the cluster center as negative examples. In simple terms, the top \(\alpha\)% of samples near the cluster center will calculate the contrastive loss according to \(\text{Cluster-NT-Xent}\), while samples that do not belong to the above range will calculate the loss according to \(\text{NT-Xent}\). When \(\alpha\) equals 100, the mathematical model is converted to \(\text{Cluster-NT-Xent}\), and when \(\alpha\) equals 0, the equation actually becomes \(\text{NT-Xent}\).

### IV. EXPERIMENTS

#### A. Data Sets

The USC-HAD [31] data set was collected on the MotionNode sensing platform and contained accelerometer and gyroscope data. This data set consists of data from 14 subjects recording 12 activities, including walking forward, walking left, walking right, going upstairs, going downstairs,
running forward, jumping, sitting, standing, sleeping, and riding the elevator up and down. All data were collected at a 100-Hz sampling rate.

The MotionSense [32] data set consists of time-series data generated by accelerometer and gyroscope sensors. An iPhone 6s was placed in the participant’s front pocket, and information was collected from the core motion framework on the IOS device using SensingKit. All data were collected at a 50-Hz sampling rate. A total of 24 participants of different genders, ages, weights, and heights performed six activities: going downstairs, going upstairs, walking, jogging, sitting, and standing in 15 trials in the same environment and under the same conditions.

The UCI-HAR [33] activity recognition data set was collected from 30 subjects who performed basic activities and postural transitions while carrying a waist-mounted smartphone with embedded inertial sensors. Six basic activities were included: 1) standing; 2) sitting; 3) lying; 4) walking; 5) going upstairs; and 6) going downstairs. Experiments captured 3-axis linear acceleration and 3-axis angular velocity at a constant 50-Hz rate using the device’s built-in accelerometer and gyroscope.

The experiments in this article will use the accelerometer and gyroscope data from the above data sets.

### B. Self-Supervised Experimental Protocol

According to work [26], [29], the USC-HAD and MotionSense data sets were segmented with 400 sample points as a sliding window with 50% overlap between two windows. According to work [28], [33], the UCI-HAR is segmented with 128 sample points as a sliding window with 50% overlap between two windows. Based on experimental protocols from previous work [28], [36], USC-HAD sensor data from subjects 11 and 12 were selected for validation, while data from subjects 13 and 14 comprised the test set. According to the splitting protocol of MotionSense and UCI-HAR, 20% of subjects are selected for the test set. Out of the remaining subjects, 20% are selected for the validation set, and the rest comprised the training set. Fivefold cross-validation was used to accomplish the aforementioned examination, and the average value was used as the experimental result.

For pretraining, ClusterCLHAR uses resampling [28] as the augmentation method and TPN [29] as the encoder, where the resampling augmentation method obtains samples at a new sampling frequency using upsampling and downsampling operations. The projection header uses three layers of MLP, all with 96 neurons, and ReLU as the activation function between two layers. The optimizer uses Adam [42] with an initial learning rate of 1e-3. The temperature coefficient is 0.1 and the model is trained for 200 epochs. The batch size is 1024 for the USC-HAD data set and 256 for the MotionSense and UCI-HAR data sets. All deep learning codes are built on the TensorFlow [43] platform. An NVIDIA GeForce RTX 3090 GPU was utilized to accelerate the training process. The contrastive loss function is Cluster-NT-Xent. The clustering method applied to USC-HAD and MotionSense is K-means [44], [45], and that applied to UCI-HAR is BIRCH [46]. The number of clustering centers defaults to the true number of classifications in the data set. The pretraining task uses the unlabeled data from the training set, and the network parameters are initialized using randomization. After pretraining, the model is thrown off the projection header, keeping only the encoder and freezing all layers, with a trainable linear classification layer added at the end of the model. In the downstream task, the optimizer uses Adam with an initial learning rate of 10. The model is trained for 200 epochs, using the mean F1-score [47] as the evaluation metric. All experiments were trained ten times, and the results were averaged. Note that the pretraining task uses the methods of Section III-A by default, and the methods of Section III-B will be shown in Section VI-C.

### C. Semi-Supervised Experimental Protocol

This part of the experiment is designed to evaluate the performance of our proposed framework under inadequate data labels. In the data preprocessing, we narrowed the sliding window to be more relevant to the situation of insufficient labels. Based on previous work [28], the USC-HAD and MotionSense data sets were segmented using a sliding window of 200 samples, with 25% overlap for USC-HAD and 12.5% overlap for MotionSense. The UCI-HAR data set maintains the settings of the self-supervised experimental protocol. To simulate the situation of insufficient labels, this article uses two training set proportion settings of 1% and 10% in the downstream task. The training set data is selected by using different random seeds.

Pretraining uses all unlabeled data from a single data set with a training batch size of 1024. The other pretraining settings are the same as the self-supervised learning experimental protocol. To more comprehensively evaluate the pretraining performance of our proposed framework, two evaluation protocols are used in the downstream task.
TABLE I
SELF-SUPERVISED LEARNING

| Method                  | Type  | Backbone | USC-HAD | MotionSense | UCI-HAR |
|-------------------------|-------|----------|---------|-------------|---------|
| 1D Conv [38]            | Sup.  |          | 49.09   | 86.66       | 79.79   |
| Transformer [27]        | Sup.  |          | 60.56   |             | 95.26   |
| TPN [29]                | Sup.  |          | 55.60   | 93.00       | 94.27   |
| Multi-task SSL [29]     | SSL   | TPN      | 45.37   | 83.30       | 80.20   |
| CAE [36]                | SSL   | CNN      | 48.82   | 82.50       | 80.26   |
| Masked Reconstruction. [37] | SSL   | Transformer | 49.31   | 88.02       | 81.89   |
| CPCCHAR [38]            | SSL   | 1D Conv  | 52.01   | 89.05       | 81.65   |
| CSSHAR [27]             | SSL   | Transformer | 57.76   |             | 91.14   |
| ClusterCLHAR (ours)     | SSL   | TPN      | 58.85   | 90.95       | 92.63   |

“Sup.” indicates supervised learning, and “SSL” indicates self-supervised learning. The evaluation criteria for the above results are F1-score. In the MotionSense and UCI-HAR datasets, 20% of the subjects’ data were used as the test set, with a five-fold cross-validation to obtain experimental results. In the USC-HAD dataset, data from subjects 13 and 14 were used as the test set.

V. EXPERIMENTAL RESULTS

A. Self-Supervised Learning

In this section, we evaluate the performance of ClusterCLHAR in the activity classification task. We compare state-of-the-art self-supervised learning work, where the training and test sets are divided in the same way. Furthermore, we quantify the performance of the self-supervised learning framework’s backbone networks on supervised learning. They are 1-D Conv, Transformer, and TPN (the backbone networks of CPCCHAR, CSSHAR, and ClusterCLHAR, respectively). This comparison shows the difference between supervised and self-supervised learning under the same backbone network with a large amount of labeled data. The experimental results are shown in Table I, and the results of the state-of-the-art work are obtained from [38] and [27].

The experimental results show that our proposed ClusterCLHAR outperforms all state-of-the-art self-supervised learning methods for three benchmark data sets. This finding demonstrates that our proposed framework can better extract data representations with unlabeled data and is effective in downstream activity recognition tasks. ClusterCLHAR outperforms supervised learning of the corresponding backbone network on the USC-HAD data set and is inferior to it on the MotionSense and UCI-HAR data sets. We speculate that this is because supervised learning performs significantly worse on the USC-HAD data set than on the other two data sets, which gives our proposed framework more space for improvement to outperform supervised learning on the USC-HAD data set. For the MotionSense and UCI-HAR data sets, we believe this is because supervised learning with a clear goal can better utilize a large amount of labeled data relative to contrastive learning with the goal of learning robust data representations. Finally, it is worth noting that while the backbone network TPN of ClusterCLHAR is inferior to the backbone network Transformer of CSSHAR in supervised learning, ClusterCLHAR outperforms CSSHAR in self-supervised learning, demonstrating the effectiveness of our suggested negative selection strategy.

B. Semi-Supervised Learning

In this section, we evaluate the performance of our proposed method in the context of contrastive learning and a small amount of data labels. We use the TPN as a benchmark for supervised learning and compare our proposed framework with three previous contrastive learning works applied to human activity recognition, SimCLR-HAR [26], MoCoHAR [28], and NNCLR [21], [25]. For a fair comparison, we use resampling data augmentation for SimCLR-HAR.

Note that our comparison focuses on the difference in model performance by the negative example definition method in contrastive learning. The experimental results are shown in Table II, and 95% confidence intervals were calculated.

The experimental results show that our proposed framework outperforms all state-of-the-art contrastive learning methods in both linear evaluation and fine-tuning for all three data sets. Moreover, the confidence intervals of our method do not overlap with those of state-of-the-art methods in most experimental settings. With 1% data labels, ClusterCLHAR outperforms the previous best contrastive learning work by 4.82% for USC-HAD, 3.3% for MotionSense, and 4.89% for UCI-HAR with linear evaluation. ClusterCLHAR outperforms the previous best work by 3.39% on USC-HAD, by 1.16% on MotionSense, and by 3% on UCI-HAR with fine-tuning. SimCLR-HAR (resampling) is most similar to our work, and the only difference is the contrastive loss function (pretraining task). Our
method outperformed SimCLR (resampling) by 5.7%, 2.93%, and 5.25% for the three data sets with linear evaluation of 1% data labels. This finding demonstrates that our approach to masking same-cluster negative examples in the contrastive loss function has a positive impact on the downstream classification task and effectively alleviates overclustering.

With the 1% labeled data, our method performs better than supervised learning in both linear evaluations and fine-tuned evaluations for the USC-HAD and MotionSense data sets. However, our approach achieves competitive performance with supervised learning on UCI-HAR possibly because under supervised learning the model is easier to train with the UCI-HAR data set relative to other data sets. This phenomenon compresses the improvement space of contrastive learning, rendering contrastive learning models without an explicit task insignificantly worse than supervised learning with an explicit classification task.

Table II

|          | Linear. | Fine. | Linear. | Fine. |
|----------|---------|-------|---------|-------|
| USC-HAD  | 70.23   | 85.93 |         |       |
| TPN (Sup.) [29] | [99.11, 71.36] | [85.51, 86.36] | [89.74, 83.32] | [93.85, 90.32] |
| SimCLR HAR [26] | 64.84   | 63.82 | 53.16   | 83.32 |
| SimCLR HAR (resampling) [28] | [44.12, 45.56] | [62.28, 65.35] | [52.62, 53.71] | [82.84, 83.82] |
| MoCoHar [28] | 71.48   | 72.87 | 83.21   | 85.30 |
| MoCoHar (resampling) [28] | [69.90, 73.06] | [71.47, 74.26] | [83.30, 83.71] | [85.00, 85.99] |
| NCCLR [25] | 68.60   | 69.00 | 78.51   | 85.84 |
| NCCLR (ours) | [67.66, 69.55] | [67.87, 70.52] | [78.21, 78.81] | [85.65, 86.02] |
| ClusterCLR HAR | 72.36   | 74.70 | 83.81   | 87.07 |
| ClusterCLR HAR (ours) | [70.97, 73.74] | [73.84, 75.56] | [83.41, 84.21] | [86.65, 87.49] |

The percentages in the table header indicate the proportion of the training set in the downstream task, and the remaining data are used as the test set. “Linear” indicates the linear evaluation and “Fine” indicates the fine-tuned evaluation. “Sup” indicates the performance of the backbone network under supervised learning. Pretraining uses all unlabeled data. The evaluation criterion for the above results is the mean F1-Score. The value inside the square brackets indicates a 95% confidence interval.

VI. ABLATION STUDIES

In this section, we perform ablation studies to investigate the impact of certain model details on the final performance. Suggestions for selecting hyperparameters based on relevant experimental results are also presented. The experiments in this section follow the semi-supervised experimental protocol, using 1% labeled data and fine-tuned evaluation by default.

A. Clustering Methods

The clustering results are critical to the performance of ClusterCLHAR. Assuming that the clustering results are close to or equal to the true classification, the contrastive learning model will encode more significant differences in the samples of different classes to better serve the downstream task. Here, we discuss the impact of using different clustering methods and corresponding clustering similarity measures on the final model performance when performing negative selection. The clustering methods applied in this section are K-means [44], [45], DBSCAN [48], hierarchical clustering [49], and BIRCH [46]. The number of true classifications is employed as the cluster number by default. The experimental results are shown in Table III.

Table III

| Model | Metric | USC-HAD | MotionSense | UCI-HAR |
|-------|--------|--------|-------------|---------|
| K-means | euclidean | 78.09 | 87.35 | 86.03 |
| BIRCH | euclidean | 68.41 | 87.97 | 90.53 |
| Hierarchical | cosine | 76.31 | 86.33 | 89.19 |
| DBSCAN | euclidean | 72.08 | 86.21 | 88.58 |
| MoCoHar | cosine | 71.82 | 85.95 | 86.10 |

“Metric” denotes the similarity measure used by the clustering method, where “euclidean” denotes the Euclidean distance and “cosine” denotes the cosine similarity.

The experimental results show that the best performing clustering method for the USC-HAD data set is K-means. K-means has a large gap with other methods, so it is chosen as the primary clustering method for this data set. The best performing clustering method on the MotionSense data set is Birch. However, K-means was chosen as the primary clustering method for this data set because it performs similarly to Birch and follows Occam’s Razor. The best-performing clustering method on the UCI-HAR data set is BIRCH, which has a larger gap with other methods, so it is chosen as the primary clustering method for this data set.
WANG et al.: NEGATIVE SELECTION BY CLUSTERING FOR CONTRASTIVE LEARNING 10841

The number in the table header indicates the true number of categories in the dataset. “True” means that the clustering method uses the true number of categories in the dataset for the number of clusters.

The experimental results show that the difference between the best performance and worst performance with different cluster numbers for the three data sets is 4.59%, 2.03%, and 3.53%. This result demonstrates that the choice of the number of clusters for the same clustering method has a significant impact on the performance of the contrastive learning model. This is because the number of clusters will directly affect the number of negative example pairs, which will impact the model training. In addition, we discover that the model performs better when the number of clusters is close to or slightly larger than the true number of classifications and worse when the number of clusters is less than the true number of classifications. For this reason, we can conclude that when pretraining the unknown classification data set for contrastive learning, we can choose the number of clusters that is slightly larger than the estimated number of classifications.

### C. Influence of the Hyperparameter $\alpha$

In this section, we implement the method of Section III-B to control the proportion of samples using our negative selection strategy by setting the hyperparameter $\alpha$. We will discuss the performance for the downstream activity recognition task with different $\alpha$. The difference between Cluster-NT-Xent ($\alpha = 100$) and NT-Xent ($\alpha = 0$) is shown in this experiment. Here, we set two cluster numbers: 1) the correct classification number and 2) 16. The experimental results are shown in Table V.

Overall, regardless of the number of clusters that is chosen, the model performs better with a large $\alpha$. The more closely the idea of Cluster-NT-Xent is followed, the better the performance. However, when all samples are used with our negative selection strategy ($\alpha = 100$), the performance is not optimal in some cases, which proves that the method of reducing the number of samples using the negative selection strategy plays a role. In other words, this method mitigates the negative impact on the model caused by clustering results that differ from the true classification. In summary, when pretraining with a new data set, we recommend that the hyperparameter $\alpha$ be set to a value close to 100.

### D. Batch Size and Epochs

This section explores the impact of using different batch sizes and epochs on model performance in the pretraining of contrastive learning. The clustering method uses the conclusions of Section VI-A, and the number of clusters uses the true classification number. The experimental results are shown in Fig. 6.

The experimental results show that the three data sets have different sensitivities for different batch sizes and epochs, and this changing trend is not easily captured. The model may overfit as the epoch increases when the batch size is determined. The selection of different batch sizes can also yield significant differences in model performance under deterministic epochs. As the batch size increases, the number of negative pairs also increases, which increases the difficulty of clustering. We speculate that such an unstable trend is that the batch size and epoch affect the performance of the clustering, which in turn affects the performance of the overall model. For this reason, the batch size and training epoch should be carefully chosen when pretraining other data sets.

### E. t-SNE

To visualize the effect of the pretraining task, we perform t-SNE [34], [35] dimensionality reduction on the output of ClusterCLHAR (Cluster-NT-Xent) and SimCLR (NT-Xent). On the MotionSense data set, we randomly select 1000 data samples to obtain 96-D representations (output of the projection header) by pretraining the model with contrastive learning. For these representations using the t-SNE method to reduce the dimensionality to 2-D, the effect is obtained, as shown in Fig. 7.

As observed in the figure, the NT-Xent distribution is rather chaotic, approximating that each sample is divided into a single class [similar to Fig. 1(a)]. The above situation is consistent with the idea of instance discrimination. Our proposed
Cluster-NT-Xent can better aggregate the same class of representations, which is very popular for downstream classification tasks. This finding again demonstrated that the introduction of clustering methods in the contrastive loss function could help the model present excellent performance in the downstream activity recognition task and effectively alleviate overclustering. The phenomenon that sample representations of the same class are more similar under the latent space can explain why our proposed method works.

### F. Training Time

Here, we explore the time required for the pretraining of several contrastive learning frameworks. We follow the semi-supervised experimental setup to count the time needed to train for one epoch on the USC-HAD data set. In addition, the corresponding backbone networks and the number of trainable parameters are presented. Code running on an Intel i9-10900KF CPU and an NVIDIA GeForce RTX 3090 GPU. The clustering method is implemented by scikit-learn, and scikit-learn-intelex is used to accelerate the clustering process. The experimental results are shown in Table VI.

The above contrastive learning frameworks all use the resampling augmentation method. “Backbone” denotes the encoder used by contrastive learning frameworks, and “Param(1000)” denotes the number of trainable parameters of the model, in thousands. “Time” indicates the time required to train the model for one epoch, in seconds.

From the experimental results, it can be seen that our suggested technique has the largest time consumption under the TPN backbone, which is because ClusterCLHAR requires the additional computation of clustering in comparison with SimCLR HAR and NNCLR. In addition, the backbone network...
has a great impact on the pretraining time consumption. For example, the only difference between SimCLRHAR and CSSHAR is the backbone network, but the difference in time consumption between the two is huge because the GPU does not accelerate the Transformer as well as the CNN. Through the above analysis, we found that time consumption is one of our shortcomings, and we will bridge this gap in terms of backbone networks and clustering optimization.

VII. CONCLUSION

To alleviate overclustering and avoid distancing the same-class sample representations, we propose a contrastive learning framework named ClusterCLHAR to select negatives by clustering in HAR. It outperforms the contrastive learning framework applied to HAR with the pretraining task of instance discrimination. In addition, ClusterCLHAR also outperforms all state-of-the-art work on self-supervised learning and semi-supervised learning for activity recognition tasks. The effectiveness of our proposed method in alleviating overclustering is demonstrated in the t-SNE visualization experiment. This article concludes with a detailed discussion of the impact of certain model details. Suggestions for selecting hyperparameters are presented based on the relevant experimental results.

Our proposed framework has some shortcomings, such as limited improvement over supervised learning with all the data labels and longer training time. In addition, we indicated that the performance of clustering can easily affect the final performance of the model in the discussion of model details. Based on the above issues, we decided to optimize the definition of same-cluster samples in the contrastive learning framework. Real-life applications of the contrastive learning framework will also be investigated.

REFERENCES

[1] P. Rashidi and D. J. Cook, “Keeping the resident in the loop: Adapting the smart home to the user,” IEEE Trans. Syst., Man, Cybern. A, Syst. Humans, vol. 39, no. 5, pp. 949–959, Sep. 2009.
[2] E. Park, Y. Cho, J. Han, and S. J. Kwon, “Comprehensive approaches to user acceptance of Internet of Things in a smart home environment,” IEEE Internet Things J., vol. 4, no. 6, pp. 2342–2350, Dec. 2017.
[3] L. Gutiérrez-Madrid, L. La Blunda, M. F. Wagner, and I. Medina-Bulo, “Test event generation for a fall-detection IoT system,” IEEE Internet Things J., vol. 6, no. 4, pp. 6642–6651, Aug. 2019.
[4] S. Kianoush, S. Savazzi, F. Vicentini, V. Rampa, and M. Giussani, “Device-free RF human body fall detection and localization in industrial workplaces,” IEEE Internet Things J., vol. 4, no. 2, pp. 351–362, Apr. 2017.
[5] S. Patel, H. Park, P. Bonato, L. Chan, and M. Rodgers, “A review of wearable sensors and systems with application in rehabilitation,” J. Neuroeng. Rehabil., vol. 9, no. 1, pp. 1–17, 2012.
[6] X. Zhou, W. Liang, K. I.-K. Wang, H. Wang, L. T. Yang, and Q. Jin, “Deep-learning-enhanced human activity recognition for Internet of Healthcare Things,” IEEE Internet Things J., vol. 7, no. 7, pp. 6429–6438, Jul. 2020.
[7] W. Zhang, T. Zhu, C. Yang, J. Xiao, and H. Ning, “Sensors-based human activity recognition with convolutional neural network and attention mechanism,” in Proc. 11th Int. Conf. Softw. Eng. Serv. Sci. (ICSESS), 2020, pp. 158–162.
[8] K. Chen, L. Yao, D. Zhang, X. Wang, X. Chang, and F. Nie, “A semisupervised recurrent convolutional attention model for human activity recognition,” IEEE Trans. Neural Netw. Learn. Syst., vol. 31, no. 5, pp. 1747–1756, May 2020.
[9] H. Zhang, Z. Xiao, J. Wang, F. Li, and E. Szczepicki, “A novel IoT-perceptive human activity recognition (HAR) approach using multthead convolutional attention,” IEEE Internet Things J., vol. 7, no. 2, pp. 1072–1080, Feb. 2020.
[10] Y. Zhao, Q. Li, F. Farha, T. Zhu, L. Chen, and H. Ning, “Indoor activity recognition by using recurrent neural networks,” in Cyberpace Data and Intelligence, and Cyber-Living, Syndrome, and Health. Singapore: Springer, 2019, pp. 205–215.
[11] K. Chen, D. Zhang, L. Yao, B. Guo, Z. Yu, and Y. Liu, “Deep learning for sensor-based human activity recognition: Overview, challenges, and opportunities,” ACM Comput. Surv., vol. 54, no. 4, pp. 1–40, 2021.
[12] P. H. Le-Khac, G. Healy, and A. F. Smeaton, “Contrastive representation learning: A framework and review,” IEEE Access, vol. 8, pp. 193907–193934, 2020.
[13] P. Bachman, R. D. Hjelm, and W. Buchwalter, “Learning representations by maximizing mutual information across views,” in Proc. Int. Conf. Adv. Neural Inf. Process. Syst., vol. 32, 2019, pp. 15535–15545.
[14] A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and P. Makedon, “A survey on contrastive self-supervised learning,” Technologies, vol. 9, no. 1, p. 2, 2020.
[15] W. Falcon and K. Cho, “A framework for contrastive self-supervised learning and designing a new approach,” 2020, arXiv:2009.00104.
[16] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 9729–9737.
[17] X. Chen, H. Fan, R. Girshick, and K. He, “Improved baselines with momentum contrastive learning,” 2020, arXiv:2003.04297.
[18] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” in Proc. Int. Conf. Mach. Learn., 2020, pp. 1597–1607.
[19] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton, “Big self-supervised models are strong semi-supervised learners,” in Proc. Int. Conf. Adv. Neural Inf. Process. Syst., vol. 33, 2020, pp. 22243–22255.
[20] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, “Unsupervised feature learning via non-parametric instance discrimination,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 3733–3742.
[21] D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet, and A. Zisserman, “With a little help from my friends: Nearest-neighbor contrastive learning of visual representations,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2021, pp. 9588–9597.
[22] S. A. Kooppayegani, A. Tejankar, and H. Pirsiavash, “Mean shift for self-supervised learning,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2021, pp. 10326–10335.
[23] G. Wang, K. Wang, G. Wang, P. H. S. Torr, and L. Lin, “Solving inefficiency of self-supervised representation learning,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2021, pp. 9505–9515.
[24] J. Robinson, C.-Y. Chuang, S. Sra, and S. Jegelka, “Contrastive learning with hard negative samples,” in Proc. Int. Conf. Learn. Represent. (ICLR), 2021, pp. 1–9.
[25] H. Qian, T. Tian, and C. Miao, “What makes good contrastive learning on small-scale wearable-based tasks?” in Proc. 28th ACM SIGKDD Conf. Knowl. Discov. Data Min., 2022, pp. 3761–3771.
[26] C. I. Tang, I. Perez-Pozuelo, D. Spathis, and C. Mascolo, “Exploring contrastive learning in human activity recognition for healthcare,” in Proc. Mach. Learn. Mobile Health Workshop NeuIPS, 2020, p. 1.
[27] B. Khaertdinov, E. Ghaleb, and S. Asteriadis, “Contrastive self-supervised learning for sensor-based human activity recognition,” in Proc. IEEE Int. Joint Conf. Biometrics (IJCB), 2021, pp. 1–8.
[28] J. Wang, T. Zhu, J. Gan, L. L. Chen, H. Ning, and Y. Wan, “Sensor data augmentation by resampling in contrastive learning for human activity recognition,” IEEE Sensors J., vol. 22, no. 3, pp. 22994–23008, Dec. 2022.
[29] A. Saeed, T. Ozccebi, and J. Lukkien, “Multi-task self-supervised learning for human activity detection,” Proc. ACM Interact. Mobile Wearable Ubiquitous Technol., vol. 3, no. 2, pp. 1–30, 2019.
[30] F. J. Ordóñez and D. Roggen, “Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition,” Sensors, vol. 16, no. 1, p. 115, 2016.
[31] M. Zhang and A. A. Sawchuk, “USC-HAD: A daily activity dataset for ubiquitous activity recognition using wearable sensors,” in Proc. ACM Conf. Ubiquitous Comput., 2012, pp. 1036–1043.
[32] M. Malekzadeh, R. G. Clegg, A. Cavallaro, and H. Haddadi, “Protecting sensory data against sensitive inferences,” in Proc. 1st Workshop Privacy Des. Distrib. Syst., 2018, pp. 1–6.
[33] D. Anguita, A. Orio, L. Oneto, X. P. Perez, and J. L. R. Ortiz, “A public domain dataset for human activity recognition using smartphones,” in Proc. 21st Int. Eur. Symp. Artif. Neural Netw. Comput. Intell. Mach. Learn., 2013, pp. 437–442.
[34] L. Van der Maaten and G. Hinton, “Visualizing data using t-SNE,” J. Mach. Learn. Res., vol. 9, no. 11, pp. 2579–2605, 2008.
[35] L. Van Der Maaten, “Accelerating t-SNE using tree-based algorithms,” J. Mach. Learn. Res., vol. 15, no. 1, pp. 3221–3245, 2014.
[36] H. Haresamudram, D. V. Anderson, and T. Plötz, “On the role of features in human activity recognition,” in Proc. 23rd Int. Symp. Wearable Comput., 2019, pp. 78–88.
[37] H. Haresamudram et al., “Masked reconstruction based self-supervision for human activity recognition,” in Proc. Int. Symp. Wearable Comput., 2020, pp. 45–49.
[38] A. Van den Oord, Y. Li, and O. Vinyals, “Representation learning with contrastive predictive coding,” 2018, arXiv:1807.03748.
[39] D. Sculley, and O. Maimon, “Clustering methods,” in Data Mining and Knowledge Discovery Handbook. Boston, MA, USA: Springer, 2005, pp. 321–352.
[40] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in Proc. Int. Conf. Learn. Represent., 2015, pp. 1–15.
[41] M. Abadi et al., “TensorFlow: Large-scale machine learning on heterogeneous distributed systems,” 2016, arXiv:1603.04467.
[42] J. MacQueen et al., “Some methods for classification and analysis of multivariate observations,” in Proc. 5th Berkeley Symp. Math. Stat. Probab., vol. 1. Oakland, CA, USA, 1967, pp. 281–297.
[43] D. Sculley, “Web-scale k-means clustering,” in Proc. 19th Int. Conf. World Wide Web, 2010, pp. 1177–1178.
[44] T. Zhang, R. Ramakrishnan, and M. Livny, “BIRCH: An efficient data clustering method for very large databases,” ACM SIGMOD Rec., vol. 25, no. 2, pp. 103–114, 1996.
[45] D. M. W. Powers, “Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation,” 2020, arXiv:2010.16001.
[46] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in Proc. KDD, 1996, pp. 226–231.
[47] S. Dasgupta and P. M. Long, “Performance guarantees for hierarchical clustering,” J. Comput. Syst. Sci., vol. 70, no. 4, pp. 555–569, 2005.