ColloSSL: Collaborative Self-Supervised Learning for Human Activity Recognition

YASH JAIN*, Georgia Tech, United States of America
CHI IAN TANG*, University of Cambridge, United Kingdom
CHULHONG MIN, Nokia Bell Labs, United Kingdom
FAHIM KAWSAR, Nokia Bell Labs, United Kingdom
AKHIL MATHUR, Nokia Bell Labs, United Kingdom

A major bottleneck in training robust Human-Activity Recognition models (HAR) is the need for large-scale labeled sensor datasets. Because labeling large amounts of sensor data is an expensive task, unsupervised and semi-supervised learning techniques have emerged that can learn good features from the data without requiring any labels. In this paper, we extend this line of research and present a novel technique called Collaborative Self-Supervised Learning (ColloSSL) which leverages unlabeled data collected from *multiple* devices worn by a user to learn high-quality features of the data. A key insight that underpins the design of ColloSSL is that unlabeled sensor datasets simultaneously captured by multiple devices can be viewed as natural transformations of each other, and leveraged to generate a supervisory signal for representation learning. We present three technical innovations to extend conventional self-supervised learning algorithms to a multi-device setting: a *Device Selection* approach which selects positive and negative devices to enable contrastive learning, a *Contrastive Sampling* algorithm which samples positive and negative examples in a multi-device setting, and a loss function called *Multi-view Contrastive Loss* which extends standard contrastive loss to a multi-device setting. Our experimental results on three multi-device datasets show that ColloSSL outperforms both fully-supervised and semi-supervised learning techniques in majority of the experiment settings, resulting in an absolute increase of upto 7.9% in F_1 score compared to the best performing baselines. We also show that ColloSSL outperforms the fully-supervised methods in a low-data regime, by just using one-tenth of the available labeled data in the best case.

CCS Concepts: • **Computing methodologies** → **Unsupervised learning**; • **Hardware** → **Sensor applications and deployments**; • **Human-centered computing** → **Ubiquitous computing**.

Additional Key Words and Phrases: Human Activity Recognition, Self-Supervised learning, Contrastive Learning

ACM Reference Format:
Yash Jain, Chi Ian Tang, Chulhong Min, Fahim Kawsar, and Akhil Mathur. 2022. ColloSSL: Collaborative Self-Supervised Learning for Human Activity Recognition. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 1, Article 1 (January 2022), 28 pages. https://doi.org/10.1145/1122445.1122456

*Ordered alphabetically, equal contribution. The work was done when the authors were visiting researchers at Nokia Bell Labs, United Kingdom.

Authors' addresses: Yash Jain, yashjain@gatech.edu, Georgia Tech, United States of America; Chi Ian Tang, cit27@cam.ac.uk, University of Cambridge, United Kingdom; Chulhong Min, chulhong.min@nokia-bell-labs.com, Nokia Bell Labs, United Kingdom; Fahim Kawsar, fahim.kawsar@nokia-bell-labs.com, Nokia Bell Labs, United Kingdom; Akhil Mathur, akhil.mathur@nokia-bell-labs.com, Nokia Bell Labs, United Kingdom.

2474-9567/2022/1-ART1 $15.00
https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

The adoption of human activity recognition (HAR) applications in mobile and wearable devices has increased significantly over the last few decades, owing to the advancements in computational models that process raw sensor data to infer human activities. Typically these computational models are trained using supervised learning, i.e., it requires a set of labeled data samples to train the models. More recently, with the popularity of data-hungry deep learning models in HAR [19, 31, 45], the need for large-scale labeled training data has become more pronounced. However, prior works have highlighted that collecting large-scale labeled HAR data is cumbersome especially outside of laboratory settings [9]. One key bottleneck for labeling HAR datasets is that sensor streams (e.g., accelerometer traces) are not easy to interpret by visual inspection, which makes any post-hoc labeling efforts non-trivial. A direct consequence of this challenge is that HAR data collection efforts [2, 35, 43, 44] are usually done at a small-scale, in a controlled or semi-controlled setting, and mostly involve less than 50 participants, thereby resulting in models which do not generalize in real-life situations.

In contrast to the challenges involved in data labeling, the collection of *unlabeled* HAR data is much easier due to the ubiquity of senor-enabled devices (e.g., smartphone, wearables) in our daily lives. As a result, machine learning approaches which utilize unlabeled data during training have been gaining prominence in the HAR literature. One of the most exciting paradigm in this direction has been self-supervised learning (SSL) [26], where the core idea is to leverage the inherent structure present in the (unlabeled) input data to derive a supervisory signal. Typically it is done by defining a *Pretext Task*, wherein a set of pre-defined transformations are applied to the input data and a deep neural network is trained to predict those transformations in the data. For example, in image datasets, the Pretext task could involve rotating an input image by 60°, and the deep neural network is trained to learn that the original and rotated image share the same embedding. We usually are not interested in the accuracy of the model on the Pretext task. Rather, we care about the intermediate representations (or features) learned by the model, with the expectation that these features will carry good semantic or structural information about the signal, which can be useful for various downstream tasks.

In recent years, we have seen some exciting explorations of self-supervised learning for activity recognition with inertial sensors. Saeed et al. [46] explored SSL in a multi-task setting wherein they proposed eight different transformations for accelerometer data, and trained a feature extractor to predict whether each transformation has been applied or not. In doing so, they found that the feature extractor managed to learn the inherent structure in the data and generate good features. Tang et al. [57] extended this idea by applying teacher-student training and showed performance gains over prior work. Haresamudram et al. [21] investigated the idea of masking sensor data at certain timesteps and training the feature encoder to reconstruct the missing data. More recently, Haresamudram et al. [22] also proposed Contrastive Predictive Coding which leverages the long-term temporal structure of sensor data streams for self-supervised learning. SSL has also been used in the context of change point detection [11] and federated learning [47].

In this work, we extend the line of research on self-supervised learning for HAR, albeit in a different context. We study a problem setting called Time-Synchronous Multi-Device System (TSMDS) which opens up an unexplored opportunity for self-supervised learning. This problem setting is inspired by the current trend of people wearing *multiple* intertial measurement unit (IMU)-enabled devices, including smartphones and consumer wearables. Studies (e.g., [49]) even estimate that by the year 2025, each person will own 9.3 connected devices on average. An example of this trend is shown in Figure 1a – here, a user is wearing multiple IMU-enabled devices which are collecting time-synchronized sensor data while the user is performing a physical activity, such as jogging.

Apart from the growing importance and practicality of this problem setting, it presents a unique opportunity for Contrastive Learning for HAR, one of the promising approaches for self-supervised learning. Contrastive learning involves comparing a data sample against its *transformed version* and other samples in the dataset. Here, in the TSMDS setting, multiple devices on a user's body are capturing the same physical phenomenon (i.e., a

(a) On-body inertial sensors (b) Acoustic sensors in a living space (c) Cameras at a traffic junction

Fig. 1. Examples of Time-Synchronous Multi-Device Systems (TSMDS) in different contexts.

user's activity) from different perspectives. For example, when a user is running, both a smartphone placed in the *thigh pocket* and a smartwatch worn on the *wrist* record sensor data for the *running activity*, but from different perspectives due to their unique placements on the body. Thus, rather than manually generating transformations of the data samples for contrastive learning, we can interpret the data from different devices in the TSMDS setting as natural transformations of each other, and leverage it to design self-supervised contrastive learning algorithms. Here, different devices collaborate in the process of self-supervised learning; hence we call this approach **Collaborative Self-Supervised Learning** (ColloSSL).

In §3, we provide a formal definition of the TSMDS setting and give a primer on contrastive self-supervised learning. In §4, we elaborate on the limitations of current contrastive learning techniques, which serve as the motivation for our research problem. In §5, we introduce novel algorithms and optimization objectives for Collaborative Self-Supervised Learning, namely Device Selection, Contrastive Sampling, and a Multi-view Contrastive Loss function. Later, we present an end-to-end semi-supervised framework which uses Collaborative Self-supervised Learning on unlabeled sensor data from multiple devices to learn a high-quality features from the data, which is followed by training an HAR classifier on a small amount of labeled data to recognize specific human activities. Finally, in §6, we compare the performance of our framework against a number of supervised and semi-supervised training baselines on three multi-device HAR datasets. Our results show that ColloSSL generally outperforms supervised training and other semi-supervised baselines in both low-data and high-data regimes. We also present insights on the feature embeddings generated by ColloSSL as well as the robustness and generalizability of ColloSSL.

Our work makes three major contributions:

- We present Collaborative Self-Supervised Learning (ColloSSL), a novel method for learning from unlabeled inertial sensor data collected from multiple devices worn by the user. Different from prior methods [46, 57], ColloSSL leverages natural transformations in the sensor datasets collected from multiple devices to perform contrastive learning, and learns a robust feature extractor for downstream HAR classification tasks.
- We introduce three key research challenges for ColloSSL and propose novel device selection and data sampling algorithms, along with a new loss function which extends contrastive learning to a multi-device setting.
- We present a thorough evaluation of ColloSSL on three multi-device HAR datasets covering both locomotion activities and complex activities of daily living. Our results show that ColloSSL outperforms both fully-supervised and semi-supervised baselines both in terms of recognition accuracy and labeled-data-efficiency.

2 RELATED WORK

In this section, we review four sets of prior works that are relevant to this paper. Specifically, we review the literature on human-activity recognition (HAR) and the emerging semi-supervised learning techniques aimed at solving the labeled data scarcity challenge in HAR. Closely tied to our work is the recent trend of investigating self-supervised learning in HAR – as such, we survey the recent ML literature on self-supervised and contrastive learning and explain how it has been applied in HAR. Finally, we also highlight the relation of our work to prior research in multi-device sensing environments.

Deep Learning for Human Activity Recognition. Human Activity Recognition is a classification task, where the labels are activities like "walking" and "running" etc. that arise naturally in day-to-day activities. The problem has been deeply studied by both signal processing researchers [13, 18–20, 41, 64], utilizing time-series data from sensors embedded in mobile devies, as well as computer vision researchers, who focused on video and camera-based solutions. For any deep learning or machine learning solution in general, the choice of feature extractor plays a crucial role. Traditionally, the solutions were focused on statistical features that have been studied by many researchers [13, 20, 41]. Later, deep convolutional and recurrent networks [18, 19, 64] were shown to be effective in learning feature extractors from labeled data.

However, the size, diversity and real-world representativeness of the datasets are crucial for satisfactory real-world performances of machine learning and deep learning solutions [34, 59, 60]. Compared to other data modalities such as images, videos and audio clips, where post-hoc annotations are often feasible, obtaining labels for sensor time-series, especially those collected in free-living environments is very difficult [5]. This is exacerbated by the variations between different use-cases and setups, which include different models or types of embedded sensors, placements of sensors, and target activity classes, making it almost infeasible to collect data which can represent all different possible setups. This leads to the paucity and limited quality of labeled data for supervision, which might have an adverse effect on performance in the real world.

Unsupervised and semi-supervised learning, which include methods that aim to make use of unlabeled data to overcome the limitations of purely-supervised methods, are some of the most popular directions of research in HAR [54, 63, 68]. Our work also builds upon this line of research on semi-supervised learning techniques, albeit in a different problem setting where we use unlabeled data obtained from multiple devices worn by a user to generate a supervisory signal for representation learning.

Self-supervised Learning for Human Activity Recognition. Self-supervised learning (SSL) has become an increasingly popular area of research in the machine learning community to minimize the reliance on labeled data for training deep learning models [6, 8, 23, 25, 29]. In this vein, a number of self-supervised learning frameworks have been proposed with their unique optimization objectives [8, 10, 17]. For example, the SimCLR [8] framework trains a feature extractor to be agnostic against transformations, by using transformed views of the same sample as positive pairs and contrasting them against other samples.

Researchers in the HAR community have recently started exploring how SSL techniques can be either extended or designed specifically for HAR tasks [46, 48, 58]. In one of the early pioneering works, Saeed et al. used the task of identifying which signal transformation has been applied to a particular data sample as a self-supervision task [46]. A set of eight signal transformations were chosen to represent signal noises, and the authors reported a performance gain by pre-training the model with this task. However, the authors focused on pre-training the models with data from similar distributions and from a single device. The signal transformation task was also adapted to train models for emotion recognition from electrocardiogram (ECG) data [50]. The overall approach follows closely to that of the previous work [46], with additional explorations on the effects on performance when using pre-training tasks of different levels of difficulties, and the relationship between the tasks used in pre-training and in application. Recently, the SimCLR framework has been applied in HAR [58]. The authors

explored a set of different combinations of transformation functions that are designed for time-series data, for training feature extractors for sensor data based on the SimCLR framework. However, this work again focused on leveraging data from a single sensor only, and the potential for extracting stronger supervisory signals from other sensors and devices was not explored. An initial attempt to leverage multiple devices for SSL has been made for visual representation [51]. It showed that time-synchronized visual representations can be used to provide a reward function for robot manipulation via reinforcement learning. One of its limitations is that it utilizes data from two camera views only, but our proposal is compatible with settings with more than two data sources, and we thoroughly explored settings with different numbers of IMU devices.

Multi-device Environments for HAR: Multi-device environments for HAR have been actively studied for the past couple decades from various angles. We briefly review the past research on multi-device environments, focusing on sensor selection and fusion for HAR. Firstly, sensor selection strategies have been proposed to maximize the system utility in body sensor networks, by dynamically selecting the best sensor based on predefined parameters of each device, such as average accuracy, resource usage, and device availability [27, 28, 36, 67]. While these strategies focus on providing better runtime system performance in a multi-device environment, they do not use the data from multiple devices to improve the accuracy of activity recognition. For this purpose, a number of deep learning based sensor fusion techniques [37, 39, 61, 65, 66] have been proposed, which train a fusion model by concatenating multiple sensor streams. While they show significant improvement in model accuracy, they assume that all sensor data streams are labeled. Instead, our work focuses on leveraging *unlabeled* data from multiple devices to learn good quality features from the data, which can later be used to train a downstream HAR model with a very small amount of labeled data. Moreover, in contrast to fusion-based approaches, we do not require the availability of all devices at inference time; the data from multiple devices is only needed during training, and inference can happen independently on each device.

3 PRELIMINARIES

In this section, we explain the TSMDS problem setting and provide a short primer on contrastive self-supervised learning.

3.1 Time-Synchronous Multi-Device Systems

We begin by providing more details on the problem setting explored in this work called Time-Synchronous Multi-Device Systems (TSMDS). It is important to note that our objective is not to claim that it is a problem setting that has not been studied before; instead, we argue that this is an interesting problem space in which self-supervised learning has not been studied. In this section, we conceptually describe this problem setting and later in §5.1 we formalize it mathematically.

In the context of human-activity recognition, the TSMDS problem setting is similar to a Body-Area Network in which multiple computing/sensor devices are worn on, affixed to or implanted in a person's body [24]. The essential characteristic of TSMDS is that all devices observe a physical phenomenon (e.g., a user's locomotion activity) *simultaneously* and record sensor data in a *time-aligned manner* (Figure 1a). Even though our work focuses on HAR using motion data, the TSMDS setting is rather generic and can be found in many other sensory applications. In a smart home (Figure 1b), multiple voice assistants (e.g., Siri in a smartphone, Alexa and Google Home in a living room) can listen to a user's speech and audio commands simultaneously. In a camera network deployed at a traffic junction (Figure 1c), multiple cameras capture the same scene from different perspectives simultaneously.

Below we state the two key assumptions we have made for exploring TSMDS setting in the context of HAR:

- We assume that all sensor devices in the multi-device system share the same sensor sampling rate, or that their data can be re-sampled to the same rate. This assumption ensures that the dimensions of the data that will be fed to the HAR model remain consistent across different devices, and it simplifies the design of the neural network architecture of the HAR model.
- By definition, we assume that multiple devices in the TSMDS setting are collecting sensor data in a time-aligned manner. Admittedly, the assumption that the sensor datasets across multiple devices are perfectly time-aligned is strong in real-world applications. There could be timestamp noise or system clock misalignment across devices, which could skew the temporal alignment of multi-device datasets. However, prior research [55] in HAR has shown timestamp noise for accelerometer and gyroscope sensors is very small, usually less than 10ms. We hypothesize that such a small noise will not degrade the performance of our solution, and empirically validate this hypothesis by showing that our approach is robust against moderate amounts of temporal misalignment between devices (§7.6).

3.2 Primer on Contrastive Self-Supervised Learning

Contrastive Learning is a type of self-supervised learning algorithm where the goal is to group similar samples in the dataset closer and dissimilar samples far from each other in the embedding space [3, 8]. In contrastive learning with unlabelled data, a data sample (called the *anchor sample*) from the training dataset is taken and often a pre-defined perturbation (e.g., rotation) is applied to generate a transformed version of the sample. During the training process, the transformed sample is considered as the *positive sample* and other randomly selected data samples from the dataset are considered as *negative samples*. All three types of data samples (anchor, positive, negative) are fed to a neural network to obtain feature embeddings. The goal of the model training is to bring the feature embeddings of the anchor and positive sample closer, while pushing the embeddings of anchor and negative samples far from each other. In this process, the model learns to extract good quality feature embeddings just using unlabeled raw data, and these embeddings could be later used for downstream classification tasks.

One of the most important factors that underpins the performance of contrastive learning is the choice of the perturbation(s) that are applied to each *anchor sample* for which we want the model to remain invariant, while remaining discriminative to other negative samples. Prior research [8, 58] has shown that the choice of perturbations chosen by the algorithm designer can have a profound impact on the performance of contrastive learning. They found that the top-1 accuracy of a model pre-trained with contrastive learning on the ImageNet dataset can drop dramatically from 56.3% (using an image crop and coloring transformation) to just 2.6% (using image rotation transformation).

4 MOTIVATION AND CHALLENGES FOR COLLABORATIVE SELF-SUPERVISED LEARNING

Instead of specifying *manual transformations* (e.g., rotation) during contrastive learning, we ask whether it is possible to leverage *natural transformations* that are already available in sensor datasets. Interestingly, the TSMDS setting presents a unique scenario where we have such natural transformations of HAR data across different devices. As shown in Figure 1a, multiple devices worn by the user are simultaneously capturing the *same physical activity* (e.g., running) from different views. As such, we can consider the dataset from different devices as natural transformations of each other. This observation can also be validated from an early seminal HAR work by Kunze and Lukowicz [30], where they argued that the accelerometer and gyroscope data collected by body-worn sensors depend on the translation and rotation characteristics of the body part where the sensor is placed. Since different body parts have different translation and rotation characteristics (e.g., wrist has a higher rotational degree of freedom than chest), it induces natural transformations in the accelerometer and gyroscope data collected from different body positions.

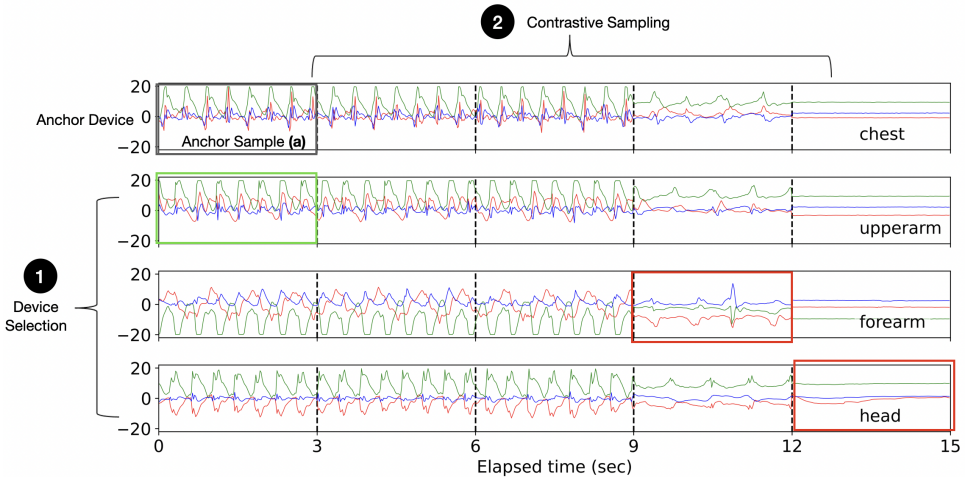

Fig. 2. Illustration of two research challenges in ColloSSL: Device Selection and Contrastive Sampling. The anchor sample is denoted by the grey rectangle. The green and red rectangles denote the positive and negative samples that are selected by our Device Selection and Contrastive Sampling algorithms described in §5.4.

In summary, the TSMDS setting allows us to capture time-aligned sensor datasets which have natural transformations across them. Our idea is to leverage these natural transformations to define a pretext task and perform contrastive learning.

Research Challenges in ColloSSL. To motivate our research challenges, we present an example illustration in Figure 2. The figure shows a 15-second trace of unlabeled accelerometer data collected simultaneously from N (=4) body-worn devices. Each accelerometer trace is segmented using a window length of 3 seconds, thereby giving us 5 windows/samples for each device. Let us say we would like to train a feature extractor for the 'chest' body position using contrastive learning. As such, the data samples from 'chest' would become the *anchor samples*. The first anchor sample from 'chest' **(*a*)** is highlighted by a grey rectangle in Figure 2. As explained above, to perform contrastive learning, we need to select appropriate positive and negative samples. Below we identify three key research questions in this direction:

- From the remaining $N-1$ devices (i.e., upperarm, forearm, head), how do we select positive and negative samples for contrastive training? Intuitively, there will be some devices whose data distribution will be too far apart from the data distribution of 'chest'. If we use these far-apart devices to obtain *positive samples* and push them closer in the embedding space to the anchor samples, it may lead to degenerate solutions for contrastive learning. Further, as the data distribution of each device changes depending on the user's activity, we need to account for these dynamic changes while selecting devices. We call this research challenge as *Device Selection* 1 and propose a data-driven strategy which uses the Maximum Mean Discrepancy (MMD) as a metric to dynamically select positive and negative devices during contrastive learning. As an illustration, we show in Figure 2 that our strategy chooses 'upperarm' as the positive device and 'forearm' and 'head' as the negative devices for the given data samples. The selection algorithm is described in detail in §5.4.1.
- In addition to *Device Selection*, we need to answer which samples from the selected devices will act as positive or negative samples. For example, if we select 'upperarm' as the positive device, which one of its 5 samples (or windows) will act as the positive sample for the anchor sample **(*a*)**? We call this challenge *Contrastive Sampling* 2 and propose the idea of using time-synchronized samples from positive devices

and time-asynchronized samples from negative devices. For example, in Figure 2, the time-synchronized sample from the positive device is highlighted with a green rectangle, whereas the time-asynchronized samples from negative devices are highlighted with red rectangles. The rationale and details behind the contrastive sampling algorithm are provided in §5.4.2.

- Finally, to enable contrastive learning in a group of devices, we need to define a new optimization objective. To this end, we propose a novel loss function called *Multi-View Contrastive Loss* which can take an arbitrary number of positive and negative samples as input and compute a loss metric, which is then optimized using stochastic gradient descent (§5.5).

5 COLLABORATIVE SELF-SUPERVISED LEARNING

In this section, we introduce our proposed approach of Collaborative Self-Supervised Learning for HAR.

5.1 Problem Statement

We formalize the TSMDS problem setting as follows: we are given D devices with time-aligned and unlabeled sensor datasets $\{\mathbb{X}_i\}_{i=1}^{D}$. Without loss of generality, we assume that the datasets are pre-segmented into T windows, as is the convention in HAR tasks. Each dataset $\mathbb{X}_i$ contains T windows $\{x_i^1, \cdots, x_i^T\}$, where x_i^j denotes a set of sensor samples from device i in window j. In general, a sensor sample could be any form of IMU measurement, e.g., 3 dimensional accelerometer data, 3 dimensional gyroscope data, 1 dimensional accelerometer norm. In our work, each sensor sample is a 6-dimensional vector, created by stacking together 3-axis of accelerometer and 3-axis of gyroscope data.

Let $D^* \in D$ be an anchor device (e.g., a smartphone) for which we want to train an HAR prediction model. Let $\mathbb{L}^* = \{(x_*^1, y_*^1) \cdots, (x_*^m, y_*^m)\}$ be a (small) pre-segmented labeled dataset from the anchor device with m windows ($m \ll T$) where x_*^j is the set of sensor samples in window j and y_*^j is the class label assigned to those samples.

Our objectives are two-fold: first, we aim to use the unlabeled datasets $\{\mathbb{X}_i\}_{i=1}^{D}$ from all the devices to train a feature extractor $f(.)$ using ColloSSL. The trained feature extractor should be able to generate high-quality feature embeddings for the anchor device data. Second, we aim to use this pre-trained feature extractor $f(.)$ to obtain feature embeddings for the labeled anchor samples x_*^j, and subsequently train an HAR classifier $g(.)$ which maps these features embeddings to the corresponding class labels y_*^j.

5.2 Solution Overview

Our proposed solution is illustrated in Figure 3 and works as follows:

1. We initialize the feature extractor $f(.)$ with random weights.
2. We sample a batch B of time-aligned unlabeled data $\{x_i^1, \cdots, x_i^B\}_{i=1}^{D}$ from all D devices.
3. **Device selection** (1) in Figure 3: Given an anchor device D^*, we first decide which of the remaining devices will provide *positive samples* and *negative samples* for contrastive learning in this batch of data. This is done through a *Device Selection* algorithm explained in §5.4.1.
4. **Contrastive sampling** (2) in Figure 3: Next, for each anchor sample, we decide out of all samples in the batch B, which specific samples from the positive and negative devices should be used for contrastive learning. While data sampling has been studied as an important aspect of SSL in the past, we present an algorithm called *Contrastive Sampling* (§5.4.2) which extends data sampling to the TSMDS setting.
5. **Multi-view contrastive learning** (3) in Figure 3: The anchor, positive and negative sample(s) are fed to the feature extractor $f(.)$ to generate feature embeddings. These feature embeddings are used to compute a Multi-view Contrastive Loss (detailed in §5.5), which pushes the positive embeddings closer to the anchor, and negative embeddings far from the anchor.

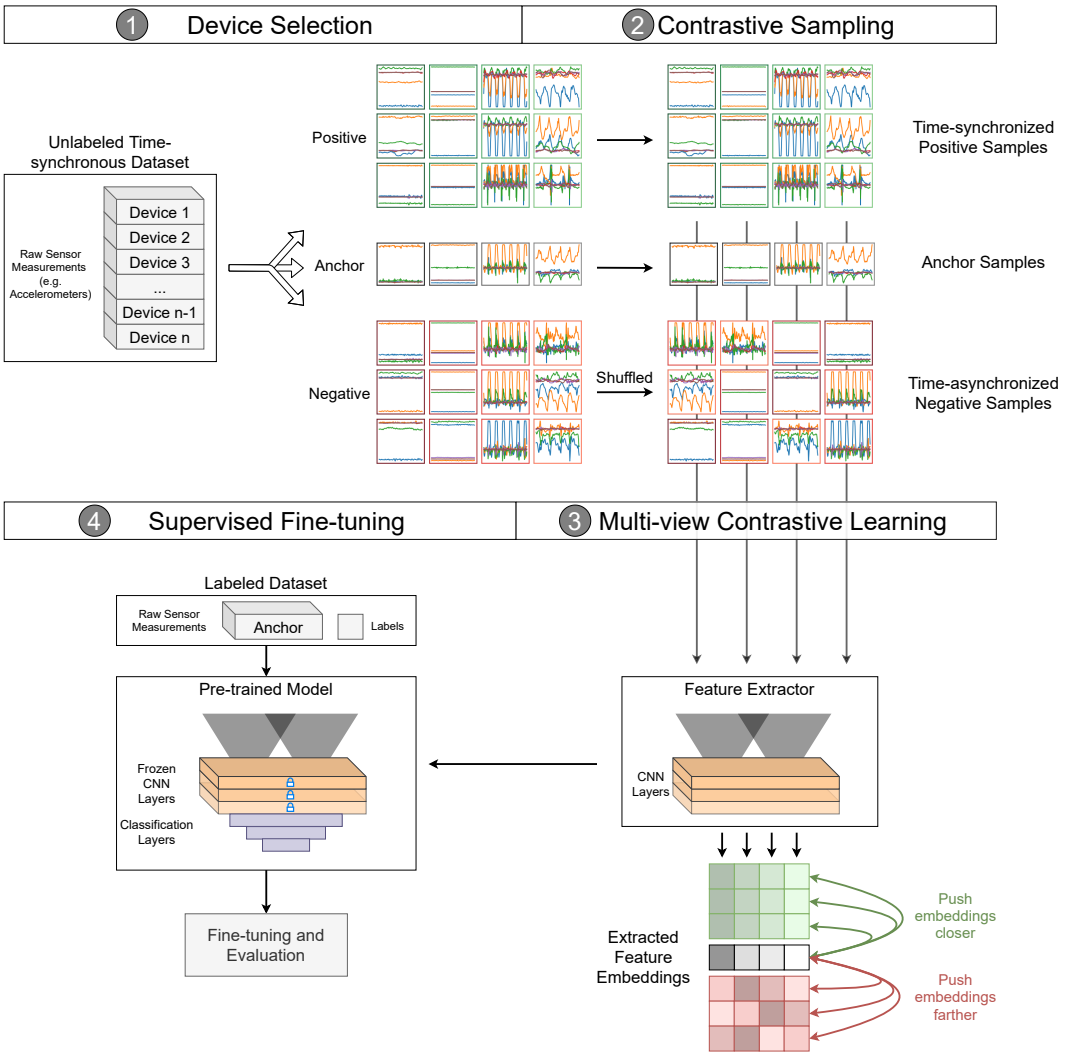

Fig. 3. Overview of Collaborative Self-Supervised Learning. Please refer to §5.2 for a detailed explanation.

6. Steps 4-5 are repeated until all anchor samples in the batch B are computed, and Steps 2-5 are repeated until the Multi-view Contrastive Loss converges. Upon convergence, we expect that $f(.)$ has learned to extract high-quality features for the anchor device.
7. **Supervised fine-tuning** 4 in Figure 3: Finally, we use the pre-trained feature extractor $f(.)$ and the labeled dataset from the anchor device ($\mathbb{L}^*$) to train an HAR classifier using supervised learning (§5.5.1).

5.3 Feature Extractor

We use a temporal (1-dimensional) convolutional neural network as the feature extractor. This design choice is inspired by prior work on self-supervised learning in HAR [46] and allows us to do a fair comparison of our

solution against the baselines by keeping the model architecture consistent across different techniques. The architecture consists of three 1D convolutional layers, with 32, 64 and 96 filters and a kernel size of 24, 16, and 8 respectively, all with stride 1. For regularization, we use Dropout between every pair of layers with a rate of 0.1, and apply L2 regularization with a regularization factor of 1e-4. Finally, the output of the last Conv1D layer is passed to a GlobalMaxPooling (1D) layer.

The input to $f(.)$ is a 2-dimensional tensor, with time on one axis and sensor data on the other. As we use both 3-axis accelerometer and 3-axis gyroscope traces as the input, the dimension of sensor data is 6. As such, given a sampling rate of 50 Hz and a window length of 2 seconds, the input tensor to the model is of dimension 100×6. The output of $f(.)$ is a feature embedding with dimension (96×1).

5.4 Device Selection and Contrastive Sampling

As explained in §4 and Figure 2, two key challenges in ColloSSL are: (1) *Device Selection*, i.e., selecting the devices from which positive and negative samples will be taken, and (2) *Contrastive Sampling*: deciding which samples from the selected devices will be used for contrastive learning.

Before we present our approach, we first explain that for a given anchor sample (x), what makes a 'good' positive (x^+) and negative (x^-) sample for contrastive learning. Recall that the objective of contrastive learning is to guide the feature extractor f to map x and x^+ closer to each other in the feature space, and x and x^- far from each other.

Goodness of positive samples. We propose that a good positive sample x^+ should have the following two characteristics:

- **(P1)** x^+ should belong to the same label/class as the anchor sample x. Because if x and x^+ are from different classes and yet the feature extractor f tries to map them closer to each other, it would lead to poor class separation and degrade the downstream classification performance.
- **(P2)** x^+ should come from a device whose data distribution has a small divergence from that of the anchor device. This is important because if the anchor device and positive device are very different in their data characteristics (e.g., wrist-worn IMU vs. chest-worn IMU), then it might affect the feature extractor in extracting meaningful invariant embeddings.

Note that in ColloSSL, we do not have access to the ground-truth labels of the data. As such, enforcing **(P1)** on x^+ may seem tricky at first. However, from the definition of TSMDS setting, we know that all devices collect the HAR data simultaneously and in a time-aligned fashion. Hence, we can naturally assume that the ground-truth labels (e.g., walking, running) are also time-aligned across devices. Therefore, if we can ensure that x and x^+ are time-aligned, it will implicitly guarantee that they have the same labels.

Goodness of negative samples. We propose that a good negative sample x^- should have the following two characteristics:

- **(N1)** x^- should be a true negative, i.e., belong to a different class than the anchor sample x. Because if x and x^- are from the same class and yet the feature extractor f tries to push them away, it could lead to poor classification performance.
- **(N2)** The most informative negative samples are those whose embeddings are initially near to the anchor embeddings, and f needs to push them far apart. In this scenario, f gets a strong supervisory signal from the data and more useful gradients during training. In the alternate scenario when negative embeddings are already far apart from anchor when the training initializes, f will receive a weaker supervisory signal from the data, which could adversely impact its convergence.

Again, due to the unavailability of class labels in ColloSSL, strictly enforcing **(N1)** is not possible. A solution is needed which can encourage this characteristic on negative samples, and minimize the possibility of x and x^- belonging to the same class.

Having defined what constitutes a good positive and negative sample, we now describe our device selection and contrastive sampling algorithms.

5.4.1 Device Selection. Our device selection algorithm is designed to increase the likelihood of selecting 'good' positive and negative samples. For brevity, we refer to the devices from which positive (or negative) samples are taken as *positive devices* (or *negative devices*). Formally, we are given a set of D devices with time-aligned and unlabeled sensor datasets $\{\mathbb{X}_i\}_{i=1}^{D}$. Let $D^* \in D$ be the anchor device. Let $D^\theta = D \setminus D^*$ be a *candidate set* of remaining devices from which we want to choose positive and negative devices.

Our device selection algorithm works as follows: first, we sample a batch of time-aligned data from the anchor device D^* and each of the device in D^θ. Let x^* and $X^\theta = \{x_i\}_{i=1}^{|D^\theta|}$ be the data batches from the anchor and the candidate devices, respectively.

We compute the pairwise Maximum Mean Discrepancy (MMD) between x^* and each of the data batches in X^θ. MMD is a distribution-level statistic to compute the distance between two data distributions; a higher MMD implies a larger distance between distributions [16]. After obtaining the batch-wise MMD scores between each pair of device batches, we use the following device selection policy:

Closest Positive. The device whose data has the least MMD distance from the anchor data is chosen as the positive device. This choice satisfies the criteria **(P2)** for selecting good positive samples and ensures that $f(.)$ can reasonably map the embeddings of the two samples closer to each other. Note that we also experimented with using more than one positive device, but found that using just one (closest) device as positive gives the best performance.

Weighted Negatives. For negative devices, we use 'all' devices from the candidate set D^θ, but their contributions during training are weighted by the inverse of their MMD distance from the anchor samples. Devices which have smaller MMD distance to anchor get higher weights during contrastive training, and devices with higher MMD distances get smaller weights. This policy serves two objectives: firstly, by assigning higher weights to devices with smaller MMD distances to the anchor, it satisfies **(N2)** and ensures that those negative samples which are closer to the anchor get more weight during training. Secondly, the use of 'all' devices as negatives serve as a hedge against the scenario when **(N1)** is violated on one device. For example, even if one device violates **(N1)** and ends up choosing x^- from the same class as x, the other devices can cover for it, and ensure that its impact on the feature extractor is minimal.

The weights assigned to each negative device $i \in D^\theta$ can be expressed as:

$$w_i = \frac{1}{MMD\,(x^*, x_i)} \tag{1}$$

The weights are further normalized by dividing each weight with the maximum weight across devices.

As an example, we apply our device selection policy to the RealWorld HAR dataset (details of the dataset are provided in §6.1). The dataset contains sensor data from 7 IMU-equipped devices: D = {chest, upperarm, forearm, thigh, shin, head, waist}. We choose 'chest' as the anchor device and obtain the pairwise MMDs between data from 'chest' and data from the remaining devices. This results in the following pairwise MMD scores: {*chest-head: 0.45, chest-waist: 0.61, chest-thigh: 0.67, chest-upperarm: 0.77, chest-forearm: 0.83, chest-shin: 1.51*}. In line with our selection algorithm, we choose *head* as the positive device and {*head, waist, thigh, upperarm, forearm, shin*} as the negative devices with weights inversely proportional to their MMD scores.

5.4.2 Contrastive Sampling. At the previous stage, we have decided which devices in D_θ will act as positive and negative. Now, we decide which data samples should be picked from each device for contrastive training.

Formally, we are given an anchor device D^*, a set of positive (D^+) and negative devices (D^-). Let $P_i = \{p_i^1, \cdots, p_i^T\}|_{i=1}^{|D^+|}$, $N_j = \{n_j^1, \cdots, n_j^T\}|_{j=1}^{|D^-|}$, $A = \{a^1, \cdots, a^T\}$ be the time-aligned data batches from the i^{th} positive, j^{th} negative, and the anchor device respectively. Here, p_i^t, n_j^t and a^t each denote a data sample at time-step t.

The objective of contrastive sampling is to select 'good' positive and negative embeddings for a given anchor sample a^t. Our sampling policy works as follows:

Synchronous Positive Samples. For a given anchor sample a^t at time-step t, we choose its time-aligned positive counterparts p_i^t as the positive samples. As explained earlier, this choice ensures that the anchor and positive samples have the same labels, and satisfies the **(P1)** criteria for good positive samples.

Asynchronous Negative Samples. A criteria for good negative samples **(N1)** is that they should belong to a different class from the anchor sample. As we do not have access to ground-truth labels during contrastive learning, it is impossible to strictly enforce **(N1)**. As a solution, we make use of the observation that negative samples which are *not time-synchronized* with the anchor are more likely to be from a different class. That is, for an anchor sample a^t at time-step t, a good negative sample would be $n^{t'} \mid t' \neq t$.

This choice however still does not guarantee that the labels at time-steps t and t' will be different; for example, a user's activity at $t = 0$ and $t' = 100$ may happen to be the same by random chance. To minimize the possibility of such cases, we use a simple trick: a large batch size of 512 is used during ColloSSL which ensures that each batch has diverse class labels in it and the possibility of a label overlap at t and t' by random chance is reduced.

5.4.3 Summary. The techniques presented in this section address the two core research challenges of ColloSSL identified in Figure 2. Using the Device Selection algorithm, we first decide which of the devices will act as positive or negative during training. Next, the Contrastive Sampling algorithm finds the 'good' positive and negative samples from the selected devices, which can be used for contrastive learning with the anchor embedding.

A curious reader may have noted that the positive and negative devices selected by our algorithm are not mutually exclusive. The positive device which has the least MMD distance from the anchor will also get selected in the set of negative devices. During Contrastive Sampling, however, the samples selected from this device will differ: when it acts as the positive device, samples which are time-synchronized with the anchor will be selected. However, when it acts as a negative device, samples which are not time-synchronized with the anchor will be selected.

5.5 Multi-view Contrastive Loss

In this section, we explain how the positive and negative samples selected from the previous step are used to train the feature extractor. Firstly, the anchor sample, positive samples(s) and negative samples are fed to the feature extractor to obtain feature embeddings. Let $\{z_i^+\}|_{i=1}^{|D^+|}$ and $\{z_j^-\}|_{j=1}^{|D^-|}$ be the selected feature embeddings from the i^{th} positive and j^{th} negative device. Let z^* be the anchor embedding.

We propose a novel loss function called Multi-view Contrastive Loss, which is inspired by the standard contrastive loss function but compatible with multiple positive and negative samples.

$$\mathcal{L}_{MCL} = -\log \frac{\sum_{i=0}^{|D^+|} \exp\left(\text{sim}\left(z^*, z_i^+\right)/\tau\right)}{\left(\begin{array}{l}\sum_{i=0}^{|D^+|} \exp\left(\text{sim}\left(z^*, z_i^+\right)/\tau\right) \\ \quad + \sum_{j=0}^{|D^-|} w_j \exp\left(\text{sim}\left(z^*, z_j^-\right)/\tau\right)\end{array}\right)} \tag{2}$$

where sim(.) denotes *cosine similarity*, w_j is weight assigned to the j^{th} negative device according to (1), and τ is a hyperparameter denoting temperature.

$\mathcal{L}_{MCL}$ is minimized for each batch of data using stochastic gradient descent. Effectively, the loss minimization during training guides the feature extractor $f(.)$ to increase the cosine similarity between anchor and positive embeddings (i.e., push them closer in the feature space), and do the opposite for anchor and negative embeddings. In doing so, $f(.)$ understands the structure of the sensor data from different devices, and learns to map raw data into good quality features, which can be useful for various downstream classification tasks.

5.5.1 Supervised Fine-tuning. Finally, after the feature extractor is trained using ColloSSL, it can be used for training down-stream HAR classification models. To this end, we follow the approach by Saeed et al. [46] of freezing the weights of the feature extractor except its last convolution layer and adding a classification head to the model. The classification head consists of a fully connected layer of 1024 hidden units with ReLU activation, followed by an output layer with the number of units equal to the number of labels. The model is then trained with a small labeled dataset $\mathbb{L}^*$ from the anchor device by optimizing the standard Categorical Cross Entropy loss.

6 EVALUATION

We evaluate ColloSSL on three multi-device HAR datasets, and compare its performance against various HAR baselines such as self-supervised learning, semi-supervised learning and fully-supervised learning. Our key results include:

- ColloSSL outperforms the fully-supervised learning in a low-data regime. In 15 out of the 18 anchor devices, ColloSSL trained with 10% or 25% of the labeled data achieves higher F_1 score than the fully-supervised model trained with 100% labeled data.
- ColloSSL also outperforms various HAR baselines in terms of recognition performance. When the same amount of labeled data is used, ColloSSL has an absolute increase of 7.9% in the F_1 score, compared to the best performing baseline.
- Through visualization of t-SNE plots and saliency maps, we show that ColloSSL generates well-separable and meaningful feature embeddings across classes.
- ColloSSL is robust to temporal misalignment of data from multiple devices; less than ±0.006 difference in the F_1 score is observed when up to 0.5s and less than ±0.01 difference when up to 3s of time synchronization error is introduced in devices.

6.1 Experimental Setup

Datasets: For our experiments, we use three datasets for human activity recognition (HAR) which have time-aligned sensor data from multiple devices: RealWorld [56], Opportunity [44], and PAMAP2 [43] as shown in Table 1. In common, they contain 3-axis accelerometer and 3-axis gyroscope data sampled simultaneously from multiple on-body devices. The inertial sensors used in two of the datasets are also heterogeneous: the RealWorld dataset uses a Samsung Galaxy S4 smartphone and a LG G smartwatch to collect the sensor data, while the inertial sensors used in the Opportunity dataset also come from different manufacturers such as InertiaCube3 and Sun SPOT.

- **RealWorld:** The RealWorld dataset [56] contains accelerometer and gyroscope traces of 15 participants, sampled at 50 Hz simultaneously on 7 sensor devices mounted at forearm, thigh, head, upper arm, waist, chest, and shin. Each participant performed 8 activities: *climbing stairs down and up, jumping, lying, standing, sitting, running/jogging, and walking.*

Dataset	No. of devices (positions)	No. of users	No. of activities (labels)
RealWorld [56]	7 (forearm, thigh, head, upper arm, waist, chest, shin)	15	8 (stair up, stair down, jumping, lying, standing, sitting, running, walking)
Opportunity [44]	5 (back, left lower arm, right shoe, right upper arm, left shoe)	4	4 (standing, walking, sitting, lying)
PAMAP2- Locomotion [43]	3 (arm, chest, ankle)	8	4 (standing, walking, sitting, lying)
PAMAP2-ADL [43]	3 (arm, chest, ankle)	8	12 (standing, walking, sitting, lying, running, cycling, nordic walking, ascending stairs, descending stairs, vacuum cleaning, ironing, rope jumping)

Table 1. Summary of datasets used for evaluation. The RealWorld and Opportunity datasets also contain heterogeneous sensors from different manufacturers.

- **Opportunity:** The Opportunity dataset [44] consists of IMU data collected from 4 participants performing activities of daily living with 17 on-body sensor devices, sampled at 30 Hz. For our evaluation, we select five devices deployed on back, left lower arm, right shoe, right upper arm, and left shoe, and we target to detect the mode of locomotion classes, namely *standing, walking, sitting, and lying*.
- **PAMAP2 (Locomotion and ADL):** The PAMAP2 Physical Activity Monitoring dataset [43] contains data of 18 different physical activities, performed by 9 participants with 3 IMUs. The IMUs were deployed over the wrist on the dominant arm, on the chest, and on the dominant side's ankle with a sampling rate of 100 Hz. Out of 9 participants, we used the data from 8 of them, since the remaining user has data for only one activity class. We performed the evaluations using two different splits of the dataset: PAMAP2 - Locomotion, which consists of 4 locomotion activities, *standing, walking, sitting, and lying*, and PAMPA2 - ADL, which consists of 12 ADL activities: *running, cycling, nordic walking, ascending stairs, descending stairs, vacuum cleaning, ironing, and rope jumping*, along with the four locomotion activities.

Baselines: We compare ColloSSL against 6 baselines, divided in the following 4 categories:

- **Random:** In the Random baseline, we assign random weights to the feature extractor and freeze them. During the supervised fine-tuning, only the classification head is trained using labeled data from the anchor device. This baseline is used to confirm that our learning task is not so trivial that it can be solved with a random feature extractor.
- **Supervised**: To represent *supervised* learning, we devise two baselines, Supervised-single and Supervised-multi. In both baselines, the feature extractor and classifier are jointly trained using labeled data by optimizing a cross entropy loss. In Supervised-single, a separate model is trained on the labeled data of each anchor device. On the other hand, Supervised-multi trains a common model using labeled data from all devices present in the dataset.
- **Semi-supervised:** As example of semi-supervised learning, we use two AutoEncoder [4] baselines: AutoEncoder-single and AutoEncoder-multi. In both these baselines, the feature extractor acts as an 'encoder' that converts unlabeled input samples into feature embeddings. We add a separate 'decoder' neural network which does the inverse task, i.e., it tries to map the feature embeddings back to the input samples. The encoder and decoder together form the AutoEncoder (AE) and are trained by minimizing the MeanSquaredError between the input data and the reconstructed data. In AutoEncoder-single, a separate AE is trained for each anchor device, where in AutoEncoder-multi, a common AE is trained using data from all devices. After the AE training converges, we discard the decoder and use the trained encoder as our feature extractor $f(.)$. Subsequently, $f(.)$ is fine-tuned on the labeled data from the anchor device.

- **Self-supervised:** To compare the performance of ColloSSL with a state-of-the-art self-supervised learning (SSL) technique, we adopt the Multi-task SSL technique proposed by Saeed et al. [46]. Multi-task SSL learns a multi-task temporal convolutional network with unlabeled data for transformation recognition as a pretext task to learn a general feature extractor and trains a recognition model based on the feature extractor with a small labeled dataset. The fine-tuning stage aligns the model to the data distribution of a particular device, and to ensure fair comparisons across different baselines on specialized models, separate recognition models are trained for each anchor device. Please note that although there are other SSL techniques proposed for HAR, we chose [46] as a baseline, because it also applies transformations to the sensor data values, thus making it a fairer comparison against ColloSSL.

Data pre-processing and hyperparameters: The accelerometer and gyroscope traces are segmented into time windows of 3 seconds for RealWorld and 2 seconds for Opportunity and PAMAP2 datasets without any overlap. These window sizes are chosen based on prior explorations with these datasets, e.g., [7, 32]. Finally, the dataset was normalized to be in the range of -1 and 1.

Our training setup is implemented in Tensorflow 2.0. We used the TF HParams API[1] for hyperparameter tuning and arrived at the following training hyperparameters: {ColloSSL learning rate: 1e-5, fine-tuning and supervised learning rate: 1e-3, $\tau = 0.05$, batch size = 512 }.

Evaluation Protocol and Metrics: In ColloSSL, even though we use unlabeled data from multiple devices to train the feature extractor, our evaluation is always done on a single anchor device (e.g., chest-worn IMU for RealWorld). We divide the participants into multiple groups and conduct leave-one-group-out cross-validation. The number of groups of RealWorld, Opportunity, and PAMAP2 is set to 5, 4, and 4, respectively. More specifically, we train the feature extractor using unlabeled data from all groups except a *held-out* group. Thereafter, the feature extractor along with the classification head is fine-tuned on a labeled dataset from the anchor device. Finally, the fine-tuned model is tested on the data from the *held-out* group.

We use the macro F_1 score (unweighted mean of F_1 scores over all classes) as a performance metric, which is considered a reasonable evaluation strategy for imbalanced datasets [40]. The weighted F_1 score is also known to take class imbalances into account, however as argued by Plötz [40], it could inflate recognition results in the favor of majority class.

6.2 Performance of ColloSSL in a Low-data Regime

We evaluate the HAR performance of ColloSSL against baseline techniques in two aspects. First, we study whether our solution performs on-par compared to the baselines in a low-data regime, i.e., whether ColloSSL shows comparable performance even with a small amount of labeled data. Second, we investigate whether our solution outperforms the baselines in terms of the recognition accuracy, with the same data availability, i.e., when all baselines are trained with the same amount of labeled data.

To study the low-data regime, we fine-tune ColloSSL and other baselines except Supervised-single, using 10%, 25%, 50%, 75%, and 100% of the available labeled data from the anchor device. The fine-tuned models are then evaluated on the anchor device from the validation/held-out group. Supervised-single is used as a reference point, thus trained using 100% of the training data. Note that the labeled training data available in our datasets for each anchor device (on average) is as follows: RealWorld: 1027 windowed samples (approximately 51 minutes), Opportunity: 3014 samples (approximately 100 minutes), PAMAP2 - Locomotion: 1280 samples (approximately 42 minutes), and PAMAP2 - ADL: 5709 samples (approximately 190 minutes).

Table 2 shows the classification performance for the various anchor devices, averaged over all validation groups in a leave-one-group-out evaluation. More specifically, we report the minimum percentage of labeled

[1] https://www.tensorflow.org/tensorboard/hyperparameter_tuning_with_hparams

Dataset (anchor)	Supervised-single	Random	Supervised-multi	AutoEncoder-single	AutoEncoder-multi	Multi-task SSL	ColloSSL
RealWorld							
forearm	0.732 (100%)	0.253 (50%)*	0.495 (100%)*	0.723 (100%)*	0.739 (75%)	0.734 (50%)	**0.767** (25%)
head	0.643 (100%)	0.211 (25%)*	0.537 (100%)*	0.647 (100%)	0.646 (25%)	0.670 (25%)	**0.690** (10%)
shin	0.781 (100%)	0.375 (100%)*	0.628 (100%)*	0.784 (10%)	0.765 (75%)*	**0.81** (10%)	**0.81** (10%)
chest	0.715 (100%)	0.228 (50%)*	0.650 (100%)*	0.478 (75%)*	0.720 (25%)	**0.722** (10%)	0.716 (25%)
thigh	0.701 (100%)	0.283 (100%)*	0.586 (100%)*	0.695 (75%)*	0.656 (25%)*	0.675 (75%)*	0.690 (25%)*
upper arm	0.726 (100%)	0.268 (75%)*	0.595 (100%)*	0.739 (75%)	0.708 (100%)*	**0.753** (10%)	0.740 (25%)
waist	0.745 (100%)	0.297 (25%)*	0.674 (100%)*	0.582 (75%)*	0.770 (10%)	0.778 (10%)	**0.781** (10%)
Opportunity							
back	0.439 (100%)	0.164 (50%)*	0.253 (25%)*	0.446 (10%)	0.445 (50%)	0.380 (25%)*	**0.556** (10%)
lla	0.370 (100%)	0.197 (100%)*	0.398 (25%)	0.386 (25%)	0.375 (25%)	0.374 (100%)	**0.516** (10%)
left shoe	0.391 (100%)	0.164 (10%)*	0.396 (75%)	0.282 (100%)*	0.172 (25%)*	0.164 (100%)*	**0.416** (25%)
right shoe	0.378 (100%)	0.164 (10%)*	0.392 (25%)	0.265 (100%)*	0.166 (50%)*	0.183 (100%)*	**0.402** (10%)
rua	0.416 (100%)	0.164 (10%)*	0.293 (100%)*	0.447 (10%)	0.375 (10%)*	0.277 (10%)*	**0.538** (10%)
PAMAP2 - Locomotion							
ankle	0.731 (100%)	0.609 (50%)*	0.589 (10%)*	0.651 (10%)*	0.770 (10%)	0.774 (50%)	**0.784** (100%)
chest	0.654 (100%)	0.295 (50%)*	0.738 (10%)	0.669 (75%)	0.655 (10%)	0.730 (10%)	**0.741** (10%)
hand	0.723 (100%)	0.496 (25%)*	0.731 (25%)	0.723 (100%)	0.750 (10%)	**0.791** (10%)	0.740 (10%)
PAMAP2 - ADL							
ankle	0.550 (100%)	0.262 (100%)*	0.548 (100%)*	0.56 (25%)	0.489 (100%)*	0.567 (50%)	**0.578** (25%)
chest	0.640 (100%)	0.156 (100%)*	0.64 (50%)	0.623 (25%)	0.607 (75%)*	0.615 (100%)*	**0.651** (50%)
hand	0.575 (100%)	0.208 (50%)*	0.585 (25%)	0.577 (75%)	0.586 (50%)	0.596 (50%)	**0.617** (25%)

Table 2. Classification performance: average of macro F_1 scores and the minimum percentage of labeled data outperforming the fully supervised model (Supervised-single). For each anchor device, bold numbers represent the highest F_1 score, and red numbers indicate the technique which requires the least amount of labeled data to outperform Supervised-single. When a technique does not outperform Supervised-single, we denote its best-achieved performance with an asterisk.

data required by each technique to surpass the performance of Supervised-single (in parenthesis), and the corresponding macro-F_1 score averaged over all validation groups. In case a technique does not surpass the performance of Supervised-single, we report its best performing F_1 score and labeled data percentage.

Our results confirm the data-efficiency of ColloSSL. In 15 out of the 18 anchor devices, including those for ADL recognition, ColloSSL with 10% or 25% of labeled data achieves higher F_1 score than Supervised-single trained with 100% labeled data. In the remaining three cases, ColloSSL shows comparable F_1 score with 25% of labeled data when evaluated at the thigh-worn device in RealWorld (ColloSSL: 0.690, Supervised-single: 0.701), a higher F_1 score with 100% of labeled data when evaluated at the ankle-worn device in PAMAP2 (ColloSSL: 0.784, Supervised-single: 0.731), and a comparable F_1 score with 50% of labeled data evaluated at the chest-worn device in PAMAP2 - ADL (ColloSSL: 0.651, Supervised-single: 0.640).

Table 2 also shows (in red font) the technique which requires the least amount of labeled data to surpass Supervised-single. We observe that ColloSSL generally performs better compared with other semi-supervised approaches (AutoEncoder-single and AutoEncoder-multi) and self-supervised approach (Multi-task SSL). More specifically, in 13 out of 18 anchor devices, ColloSSL used the lowest percentage of labeled data across the baselines. This is remarkable considering that AutoEncoder-single, AutoEncoder-multi, and Multi-task SSL win at 5, 4, and 6 anchor devices; note that multiple winners can be chosen.

Finally, Table 2 shows (in bold font) the technique which provides the highest performance in a low-data regime. Here ColloSSL has the highest F_1 score in 14 out of 18 anchor devices across all techniques. Multi-task

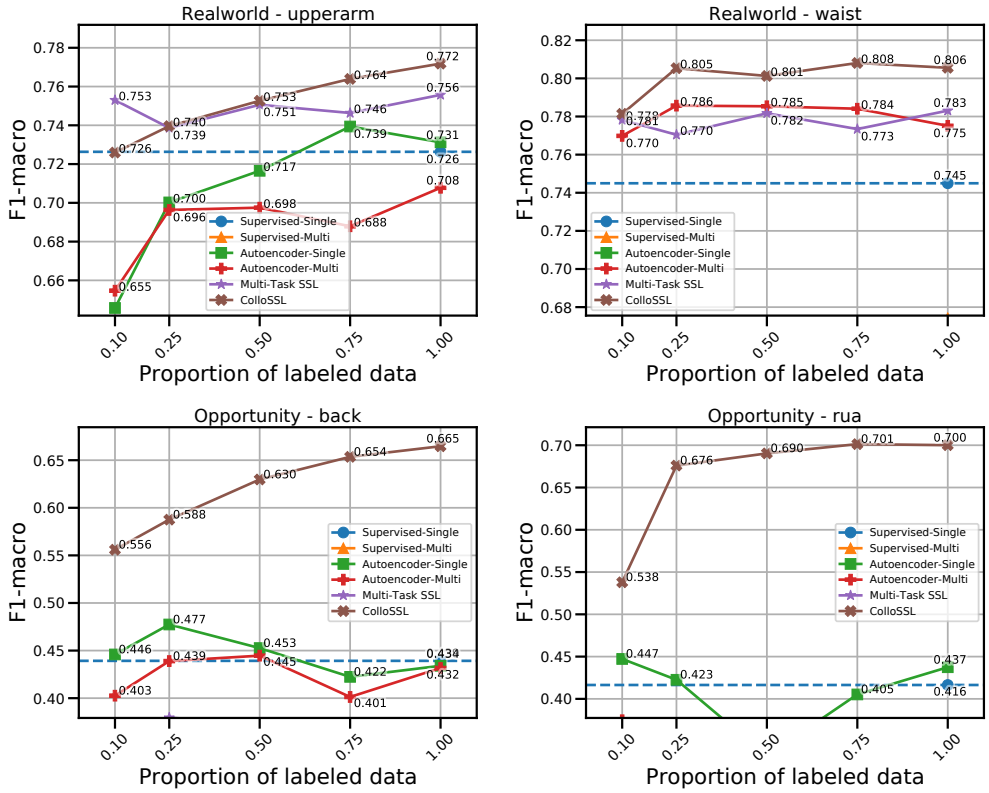

Fig. 4. Assessing the classification performance of ColloSSL and baselines across different percentages of labeled data. Note that some baselines had very poor performance and are not visible in the plot. Please refer to Table 2 for in-depth results.

SSL also outperformed the supervised baseline in most cases, and outperformed ColloSSL in 3 of the remaining scenarios. This could be attributed to the data-efficiency of self-supervised methods, and in certain scenarios, pre-text tasks based on specific data transformations could offer the right data diversity for training a feature extractor. However, for *Opportunity* dataset, there are several cases where the Multi-task SSL baseline failed to outperform the supervised baseline. One possible reason for this is that the effectiveness of SSL methods which rely on manual data transformations can be highly dependent on the dataset and the specific transformations used by the technique [8]. It is possible that the manual transformations employed by this baseline were not optimal for Opportunity, which resulted in poor recognition accuracy. Instead, ColloSSL does not define any manual transformations on the datasets and leverages natural transformations present in the data; hence overall it is more robust to dataset variations.

Figure 4 provides further insights into these findings by plotting the performance of various techniques when they trained or fine-tuned with different percentage of labeled data from the anchor device. We present two important findings. First, regardless of the percentage of labeled data used for fine-tuning, ColloSSL generally outperforms the baselines. This shows that our design of device selection, contrastive sampling, and group contrastive loss contributes to enhancing the performance of HAR. Second, we can again observe that ColloSSL

Dataset (anchor)	Random	Supervised-single	Supervised-multi	AutoEncoder-single	AutoEncoder-multi	Multi-task SSL	ColloSSL
RealWorld							
forearm	0.248 (0.028)	0.732 (0.065)	0.495 (0.039)	0.723 (0.045)	0.718 (0.064)	0.738 (0.057)	**0.774 (0.053)**
head	0.123 (0.028)	0.643 (0.055)	0.537 (0.031)	0.647 (0.071)	0.627 (0.061)	0.663 (0.026)	**0.730 (0.046)**
shin	0.375 (0.045)	0.781 (0.044)	0.628 (0.060)	0.799 (0.064)	0.761 (0.078)	0.785 (0.052)	**0.806 (0.103)**
chest	0.135 (0.030)	0.715 (0.104)	0.650 (0.054)	0.461 (0.127)	**0.729 (0.09)**	0.708 (0.061)	0.720 (0.095)
thigh	0.283 (0.024)	**0.701 (0.11)**	0.586 (0.022)	0.670 (0.061)	0.616 (0.088)	0.651 (0.120)	0.679 (0.101)
upper arm	0.126 (0.028)	0.726 (0.090)	0.595 (0.019)	0.731 (0.066)	0.708 (0.063)	0.756 (0.084)	**0.772 (0.042)**
waist	0.157 (0.049)	0.745 (0.127)	0.674 (0.042)	0.579 (0.167)	0.775 (0.051)	0.783 (0.102)	**0.806 (0.070)**
Average	0.207 (0.090)	0.720 (0.039)	0.595 (0.058)	0.659 (0.103)	0.705 (0.057)	0.726 (0.050)	**0.755 (0.044)**
Opportunity							
back	0.164 (0.010)	0.439 (0.092)	0.217 (0.012)	0.434 (0.114)	0.432 (0.107)	0.355 (0.076)	**0.665 (0.134)**
lla	0.197 (0.050)	0.370 (0.013)	0.396 (0.082)	0.458 (0.028)	0.369 (0.016)	0.374 (0.011)	**0.553 (0.018)**
left shoe	0.164 (0.009)	0.391 (0.043)	0.394 (0.046)	0.282 (0.073)	0.171 (0.010)	0.164 (0.009)	**0.443 (0.040)**
right shoe	0.164 (0.009)	0.378 (0.024)	0.354 (0.032)	0.265 (0.056)	0.164 (0.009)	0.183 (0.011)	**0.448 (0.026)**
rua	0.164 (0.009)	0.416 (0.060)	0.293 (0.068)	0.437 (0.126)	0.277 (0.058)	0.185 (0.034)	**0.700 (0.131)**
Average	0.171 (0.013)	0.399 (0.025)	0.331 (0.068)	0.375 (0.084)	0.283 (0.106)	0.252 (0.092)	**0.562 (0.107)**
PAMAP2 - Locomotion							
ankle	0.558 (0.115)	0.731 (0.100)	0.558 (0.072)	0.635 (0.012)	0.754 (0.081)	0.720 (0.095)	**0.784 (0.088)**
chest	0.160 (0.052)	0.654 (0.136)	0.680 (0.082)	0.687 (0.120)	0.639 (0.098)	0.716 (0.141)	**0.742 (0.112)**
hand	0.397 (0.180)	0.723 (0.111)	0.738 (0.092)	0.723 (0.105)	0.729 (0.088)	**0.777 (0.065)**	0.737 (0.078)
Average	0.372 (0.163)	0.703 (0.121)	0.659 (0.111)	0.682 (0.099)	0.708 (0.102)	0.738 (0.109)	**0.754 (0.097)**
PAMAP2 - ADL							
ankle	0.262 (0.038)	0.550 (0.124)	0.548 (0.135)	0.566 (0.090)	0.489 (0.151)	0.559 (0.096)	**0.646 (0.184)**
chest	0.156 (0.062)	0.640 (0.189)	0.655 (0.177)	**0.66 (0.150)**	0.606 (0.104)	0.615 (0.169)	**0.66 (0.185)**
hand	0.170 (0.030)	0.575 (0.078)	0.647 (0.090)	0.575 (0.063)	0.585 (0.095)	0.621 (0.089)	**0.664 (0.087)**
Average	0.196 (0.047)	0.588 (0.038)	0.617 (0.049)	0.601 (0.044)	0.560 (0.051)	0.598 (0.028)	**0.657 (0.008)**
Total average	0.237	0.603	0.551	0.579	0.564	0.579	**0.682**

Table 3. Comparison of classification performance (average and standard deviation of macro F_1 scores) for different anchor devices, when 100% of the labeled data is available for fine-tuning. lla: left lower arm, rua: right upper arm.

outperforms the fully-supervised model (Supervised-single) even with much less labeled data , including for PAMAP2 - ADL, which contains with more complex activities of daily living.

6.3 Comparison of ColloSSL Recognition Performance with Baselines

We now compare the classification performance of ColloSSL against various baseline techniques when sufficient labeled data is available. Here, we use 100% of the labeled training data available from the anchor device for fine-tuning ColloSSL, AutoEncoder-single, AutoEncoder-multi, and Multi-task SSL, and for training Supervised-single and Supervised-multi. Then, we evaluate these techniques on the anchor device from the held-out group. A hyperparameter search on training parameters was conducted for all pipelines to ensure optimal performance.

Table 3 shows the mean and standard deviation of macro F_1 scores for different anchor devices. On average, the results show that ColloSSL outperforms baseline techniques for all datasets we used. ColloSSL has an absolute increase of around 7.9% in the F_1 score over the average of all anchor devices across all datasets, compared to the best performing baseline, Supervised-single. We also observe that ColloSSL outperforms Multi-task SSL, a state-of-the-art self-supervised learning technique for HAR for all except one anchor device. This validates our

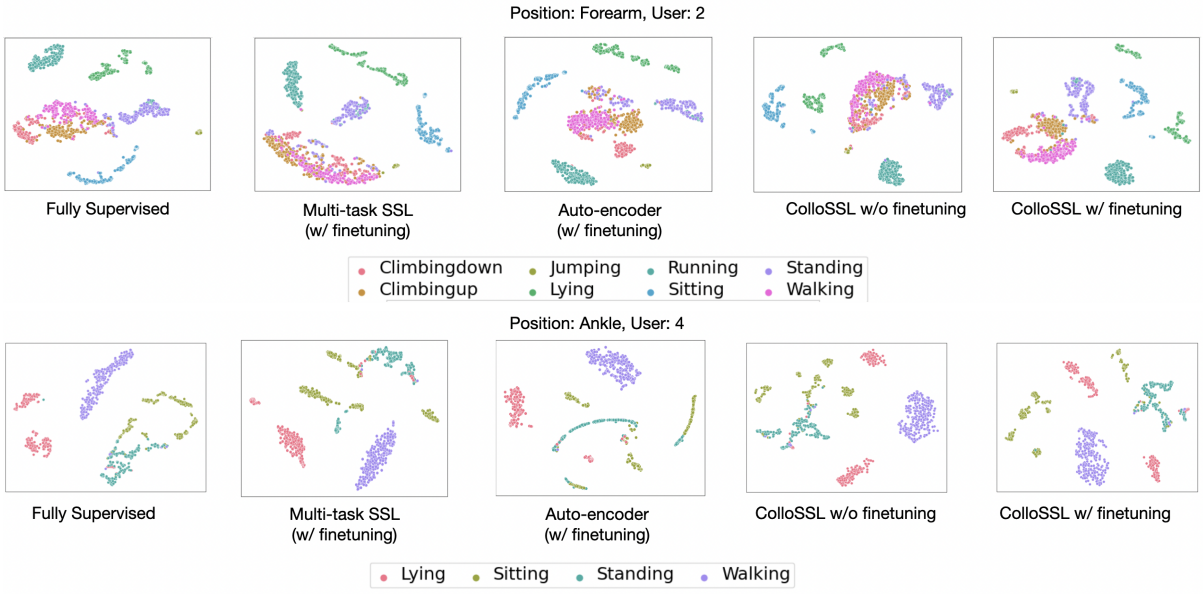

Fig. 5. t-SNE visualizations to compare the features learned by ColloSSL and various baselines.

design choice of leveraging natural transformations from the TSMDS settings for self-supervised contrastive learning. Furthermore, our method outperforms the best performing baseline by 4% in F_1 score in absolute terms in PAMAP2 - ADL, which demonstrates that our proposed method can offer performance gain in simpler locomotion recognition, as well as more complex ADL recognition.

6.4 Embedding Similarity and Data Saliency

In this section, we delve deeper to analyze the feature embeddings learned by the feature extractor and compare them between ColloSSL and Fully Supervised settings. We also present Saliency Maps to understand how the HAR models trained in these two settings are making their predictions.

Visualizing the feature space using t-SNE plots: We visualize the learned feature embeddings of ColloSSL and the baselines (Supervised-single) using t-distributed stochastic neighbor embedding (t-SNE) plots [62]. t-SNE is a statistical method that is used to explore and visualize high-dimensional data. Based on Stochastic Neighbor Embedding, it reduces the dimensionality of data in a nonlinear way and places each data point in a location in a space of two or three dimensions. Specifically, t-SNE aims to discover patterns in the data by clustering data points based on their similarity.

Using t-SNE, we project the 96-dimensional feature embeddings generated by the feature extractor onto a 2D space in the following settings: ColloSSL w/o finetuning, ColloSSL w/ finetuning, and fully-supervised, multi-task SSL and autoencoder-single. Figure 5 shows the t-SNE plots for two users and two anchor devices from the RealWorld (top) and PAMAP2 (bottom) datasets. In common, we observe that ColloSSL w/o finetuning already generates well-separable feature embeddings across classes. It implies that ColloSSL captures the semantic structure of the data very well. The class separation in the embeddings is further improved by finetuning with a small amount of labeled data as shown in ColloSSL w/ finetuning. We also observe that the nature of clustering

(a) Climbing down activity collected from a chest-work IMU (b) Walking activity collected from a waist-worn IMU

Fig. 6. Saliency map for samples of RealWorld dataset; (top) raw input signal, (middle) and (bottom) magnitude values computed from the input signal. The intensity of color indicates the impact of the region on the model prediction. We observe that ColloSSL and fully supervised model show similar patterns of the color intensity. This implies that models trained with both approaches largely focus on similar regions of the data to make their predictions.

learned with ColloSSL is largely comparable with those learned with fully-supervised model trained with 100% of the labeled data. The two baselines, autoencoder and multi-task SSL, also achieve a reasonably good cluster separation; however certain classes end up having high overlap in their features. For example, multi-task SSL finds it difficult to separate the data *Climbing up, Climbing Down, and Walking* activities in Figure 5 (top).

Visualizing the salient regions in the data using Saliency maps. For a better interpretability of our findings, we visualize saliency maps [46, 53] for two randomly selected data samples from the RealWorld dataset. A saliency map illustrates the regions of the data sample that have the most effect on a model's prediction; the parts with higher color intensity shows the regions that contribute most to the model's prediction. Our objective is to understand if the salient regions of the data remain consistent across ColloSSL and Fully-Supervised training.

Figure 6 shows the saliency maps for two randomly selected data samples from the RealWorld dataset; a sample of a *climbing down* activity collected from a chest-worn IMU in Figure 6a and a sample of a *walking* activity collected from a waist-worn IMU. We visualize the three-axis raw input data from the accelerometer and gyroscope separately in the top pane. The middle and bottom panes show the saliency maps for this input data produced by ColloSSL and Fully-supervised training for a class with the highest prediction score. For ease of understanding, we only present the magnitude values of the accelerometer and gyroscope data in the saliency maps. In the middle and bottom panes, the intensity of color indicates the impact of the region on the model prediction. The regions with strong intensity imply that they contribute to the model prediction more than those with weak intensity.

Figure 6 shows that ColloSSL and the fully-supervised model show a similar pattern in color intensity, both for accelerometer and gyroscope samples. For example, in Figure 6a, the periodic peaks in the accelerometer data on

Anchor	Closest Positve & Random Negative	Random Selection	ColloSSL
chest	0.649 (0.175)	0.631 (0.166)	**0.662 (0.185)**
ankle	0.602 (0.122)	0.553 (0.095)	**0.646 (0.184)**
hand	0.651 (0.088)	0.634 (0.085)	**0.664 (0.087)**

Table 4. Comparison of various device selection strategies in terms of performance (average and standard deviation of macro F_1 scores) for the PAMAP2 - ADL dataset.

the x-axis (blue color) are largely responsible for the model's prediction in both ColloSSL and fully-supervised settings. The takeaway from this result is that the models trained with ColloSSL and fully-supervised training largely focus on similar regions of the data to make their predictions. This confirms that ColloSSL is able to generate meaningful representations of data for the HAR classification task.

7 ANALYSIS

In this section, we present a set of ablation studies and also analyze the performance of ColloSSL under real-world constraints in sensor devices. Please note that due to space constraints, we present the results on only one dataset and a subset of anchor devices in each section, however the results also hold for other scenarios.

7.1 Analysis of the Device Selection Strategy

Our proposed device selection strategy (§5.4.1) uses the closest device (least MMD distance) to the anchor as the Positive device and all devices as Negatives, weighted by the inverse of their MMD distance to anchor. In this ablation experiment, we compare this strategy against two baselines: i) Random Selection ii) Closest Positive and Random Negative Selection. In the Random Selection strategy, we randomly pick one positive and one negative with replacement in each batch. In the Closest Positive & Random Negative strategy, we pick the closest device with the least MMD distance to anchor, but randomly choose one negative device.

Table 4 shows the experiment result on the PAMAP2 - ADL dataset with 12 ADL activities. We observe that ColloSSL outperforms the two baseline approaches. In particular, the Random Selection strategy performs the worse as it often picks positive devices which have different data distributions from the anchor device. The Closest Positive & Random Negative also has a lower performance as it does not prevent the violation of the **(N1)** criteria for negative samples as described in §5.4. This finding supports our hypothesis that using 'all' devices for negative samples serve as a hedge against the scenario when **(N1)** is violated on one of the devices.

7.2 Analysis of the Contrastive Sampling Approach

We now evaluate the effect of contrastive sampling by running an ablation on the PAMAP2-ADL dataset. We compare ColloSSL's asynchronous negative sampling against its counterpart, synchronous negative sampling. Note that, positive samples are sampled synchronously in both the settings, otherwise positive samples will violate characteristic **(P1)**. Table 5 exhibits the improvement in performance by using asynchronous negative sampling. We attribute this gain to the knowledge that synchronous samples in TSMDS setting belongs to the same class as the anchor sample; hence by negatively contrasting these samples, the feature extractor is violating **(N1)** and learns poor representations.

7.3 Analyzing the Role of Weights in Multi-view Contrastive Loss

In multi-view contrastive loss, we introduce weights as a way to differentiate between the contributions of each negative device towards the optimization objective. To investigate the effect these weights have on ColloSSL, we conducted an experiment where the weights were removed from the loss function (i.e., all devices were assigned the same weight). Table 6 shows the result of our experiment. We can observe that use of the weighted loss

Anchor	Synchronous positive and negative	ColloSSL
chest	0.6307 (0.180)	**0.662 (0.185)**
ankle	0.601 (0.101)	**0.646 (0.184)**
hand	0.639 (0.076)	**0.664 (0.087)**

Table 5. Comparison of the effect of contrastive sampling on performance (average and standard deviation of macro F_1 scores) for the PAMAP2 - ADL dataset.

Anchor	ColloSSL w/o weights	ColloSSL
chest	0.658 (0.184)	**0.662 (0.185)**
ankle	0.601 (0.112)	**0.646 (0.184)**
hand	0.636 (0.076)	**0.664 (0.087)**

Table 6. Comparison of the effect of Weights over performance (average and standard deviation of macro F_1 scores) of ColloSSL in the PAMAP2 - ADL dataset.

criterion improves the performance of ColloSSL which supports our hypothesis that weighted negatives help in pushing closer negative embeddings further away from the anchor device **(N2)** resulting in better features.

7.4 Robustness to Sensor Heterogeneity

In the TSMDS setting, devices placed at different body positions could be heterogeneous, in that they can come from different manufacturers or use different inertial sensors. In this section, we probe the robustness of ColloSSL to sensor heterogeneity by synthetically adding two types of heterogeneity in the IMU data based on prior literature [14, 15, 42].

Prior research has shown that deterministic errors in IMU sensors are the prominent causes of heterogeneity in the sensor data. Deterministic errors are caused by variations in sensor components across manufacturers, imperfections introduced in the analog circuitry of the sensor during the manufacturing process [12], or temperature differences between initial calibration and operational stages [1]. Two of the major types of deterministic errors are scale factor errors and bias errors [15].

For this experiment, we induce different scale factor and bias errors to each IMU device in the RealWorld dataset. For each device, we sample a scale factor S from a normal distribution with $\mu = 1.0$ and $\sigma = 0.05$ (low heterogeneity) and $\sigma = 0.1$ (high heterogeneity). Similarly, we sample a bias factor B from a normal distribution with $\mu = 0.0$ and $\sigma = 0.05$ (low heterogeneity) and $\sigma = 0.1$ (high heterogeneity). Following the methodology proposed in [15], the two factors are introduced to the raw datasets $\{\mathbb{X}_i\}_{i=1}^{D}$ to obtain $\mathbb{X}'_i = S \times (\mathbb{X}_i - B)$, where $\mathbb{X}'_i$ denotes the dataset with induced sensor heterogeneity for the i^{th} device. Thereafter, we run the end-to-end training pipeline of ColloSSL on the heterogeneous datasets $\{\mathbb{X}'_i\}_{i=1}^{D}$ using the same experiment protocol as previous experiments.

Our findings are shown in Table 7. For comparison, we also present the results when no additional sensor error is added to the dataset. Overall, we observe that ColloSSL is able to handle sensor heterogeneity and outperforms the fully-supervised and multi-task SSL baselines. Interestingly, we found that introducing 'low heterogeneity' to the unlabeled data improves the performance of ColloSSL and the other baselines. This finding can be explained by prior work which has shown that data augmentation based training of deep neural networks helps in learning better more generalizable features [33, 38].

7.5 Robustness to Missing Devices

In real-world scenarios, it is often the case that all the devices are not available all the time, e.g., the device runs out of battery, or a user takes off their earbuds during a conversation. This would result in having missing signal

	None			Low			High		
Anchor	Supervised single	Multi-task SSL	ColloSSL	Supervised single	Multi-task SSL	ColloSSL	Supervised single	Multi-task SSL	ColloSSL
forearm	0.732	0.738	**0.774**	0.744	0.762	**0.786**	0.73	0.74	**0.761**
shin	0.781	0.785	**0.806**	0.806	0.811	**0.8337**	0.773	0.782	**0.80**

Table 7. Performance (macro F_1 scores) for two anchor devices in the RealWorld dataset under different levels of sensor heterogeneity. 'None' denotes the case where no additional sensor error is added to the dataset.

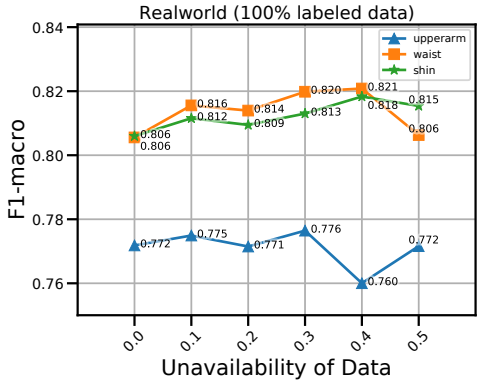

Fig. 7. Assessing the classification performance of ColloSSL across different unavailability of devices in TSMDS setting. Note that unavailability of each device (x-axis) is decided independently of each other. Please refer to § 7.5 for more details.

data from some devices in the TSMDS setting. We explore this missing data problem for ColloSSL and conduct an experiment with RealWorld dataset. While preparing the data for this experiment, we assume that the anchor device, for which we would like to learn a prediction model, is always available. For the remaining N devices, we set the unavailability of each device with the probability, p_u= {0.1, 0.2, 0.3, 0.4, 0.5}. For example, if p_u is set to 0.1, all devices will be available in the same time window with the probability of $(1-0.1)^N$. More specifically, when a device is set to unavailable in a given time window, we replace its sensor data with zeros.

Fig 7 shows the F_1-macro values with varying availability probability values. The results show that ColloSSL is robust against the changing availability of devices and we observe at most a 1% performance drop in our experiments due to device unavailability. This result can be explained by the design of device selection and weighted loss function in ColloSSL. Firstly, missing devices (i.e., devices with 0 data) will have a high MMD with the anchor device, and ColloSSL is likely to assign them as negative devices. Secondly, the contribution of these missing devices will be significantly down-weighted in the multi-view loss function as negative devices with high MMD distances get assigned smaller weights in training. Surprisingly, we also observe that ColloSSL with missing devices sometimes provides slightly higher performances than the case with full device availability. Admittedly, we do not have a good explanation for this behavior; we surmise that the neural network considers missing data as a form of noise, which might lead to an implicit training regularization that boosts performance.

7.6 Robustness to Temporal Misalignment

In §3.1, we assume that data from multiple devices in the TSMDS setting are collected in a time-aligned manner. To investigate how robust ColloSSL is to temporal misalignment between devices, we conduct an experiment with the RealWorld dataset by deliberately injecting time synchronization errors. More specifically, we shift the

Anchor device (RealWorld)	Time Synchronization error							
	0s	0.01s	0.1s	0.5s	0.75s	1.5s	2.25s	3s
waist	0.806	0.804	0.800	0.808	0.811	0.805	0.802	0.812
shin	0.806	0.809	0.809	0.805	0.807	0.813	0.811	0.815

Table 8. Comparison of classification performance (average of macro F_1 scores) for different time synchronization errors.

Device for testing	ColloSSL	ColloSSL-unseen
upper arm	0.772 (0.042)	0.764 (0.063)
waist	0.806 (0.070)	0.792 (0.072)

Table 9. Comparison of classification performance (average and standard deviation of macro F_1 scores) between ColloSSL and ColloSSL-unseen in the RealWorld dataset.

timestamps of all devices in the RealWorld dataset by 0.01, 0.1, 0.5, 0.75, 1.5, 2.25, and 3 seconds, except the anchor device.

Table 8 shows the F_1-macro values for two anchor devices, *waist* and *shin*. The results show that, for realistic, moderate time-sync errors ($\leq$ 0.5 seconds), there is no significant change in the performance of ColloSSL, i.e., within ±0.006 of the F_1-macro value. Even with high misalignment casees ($>$ 0.5 seconds), the change in the F_1-macro score is about ±0.01. We conjecture that this result is caused by a) the temporal locality of human behaviors, and b) the ability of the feature extractor $f(.)$ to ignore moderate synchronization errors.

7.7 Generalizability of the Feature Extractor

We further investigate the generalizability of ColloSSL: whether the feature extractor $f(.)$ trained using ColloSSL is transferable to new devices, i.e., the ones that do not participate in pre-training of $f(.)$. To this end, we pre-train *ColloSSL-unseen* on unlabeled data from all devices except one 'unseen' device. The pre-trained model is then fine-tuned and evaluated on the unseen device. For example, in the RealWorld dataset – if 'head' is chosen as the unseen device, we pre-train the feature extractor on rest of the devices to obtain ColloSSL-unseen. Then, we fine-tune ColloSSL-unseen with labeled data of 'head', and report the classification performance using test data of 'head'.

Table 9 compares the classification performance between ColloSSL and ColloSSL-unseen when the model is evaluated at upper arm- and waist-worn devices in the RealWorld dataset. The results show that ColloSSL-unseen shows comparable performance to ColloSSL, even though the data of the unseen device is not used for pre-training the feature extractor. The decrease of F_1 score is less than 1% in both cases. This gives us an early indication that the feature extractor, $f(.)$, trained using ColloSSL is transferable across devices and can be useful for finetuning on unseen devices. More specifically, when a new, unseen device is added to the TSMDS setting, we can reuse the pre-trained $f(.)$ and just fine-tune it using a small amount of labeled data from the new device.

8 DISCUSSION AND LIMITATIONS

In this section, we discuss the limitations of our approach, elaborate on some of the practical deployment concerns associated with our method, and highlight avenues for future research on this topic.

Training Cost of ColloSSL. Training a model using ColloSSL naturally takes more time when compared to fully-supervised learning, because of the need for pre-trainig on unlabeled data. However, since model training is currently done offline (e.g., on a server), it has no adverse implications for system resources on mobile or wearable devices. Further, ColloSSL does not impose any additional costs for data collection; in the TSMDS setting, multiple devices (e.g., smartphone, smartwatch) are anyway collecting sensor data related to a user's activity, and ColloSSL simply uses this unlabeled data to train a more accurate HAR model.

Runtime System Cost. Although ColloSSL uses data from multiple devices to train the HAR model, it is important to note that the trained model using ColloSSL only operates on a single device at runtime, similarly to any conventional HAR model. Hence, we expect that the system costs of ColloSSL, such as inference latency and energy consumption on mobile and wearable devices, are the same as an HAR model trained using supervised learning.

ColloSSL as a general framework for learning in TSMDS settings. Although we focus on applying ColloSSL to HAR with motion data, the TSMDS setting is common to other sensor modalities (e.g., audio, vision) as shown in Figure 1. To apply ColloSSL to other TSMDS settings, the technical solutions (device selection, contrastive sampling, and group contrastive loss) need to be redesigned to reflect the characteristics of sensory signals, user behavior, and environments. However, we believe the key idea of ColloSSL is still valid, which is to leverage natural transformations in the unlabeled datasets from multiple devices to generate a supervisory signal for training. In future work, we plan to explore technical solutions to extend ColloSSL to audio- and vision- based TSMDS settings.

We now discuss several limitations of our current approach.

Data Privacy. ColloSSL is designed as a collaborative learning framework, in that it requires raw data from multiple devices to train the HAR model. In practice, the sensor devices owned by a user may be from different device manufacturers, who may not be willing to offload the raw sensor data to a centralized cloud server due to privacy and commercial reasons. We envision two potential solutions to this issue: firstly, model training can be done on a trusted edge device such as a home router and it ensures that a user's data never leaves their premises. Alternatively, federated self-supervised learning approaches [52] can be explored wherein the feature extractor is trained locally on each device and only the gradients of the feature extractor are shared to a central server for aggregation.

Extension to newer SSL algorithms. This work focuses on using a contrastive learning paradigm with positive and negative samples for self-supervised learning. However, recently novel SSL methods have been proposed which do not require negative samples [10, 17] and outperform contrastive learning methods such as SimCLR. As a future work, we plan to explore such methods for self-supervised learning in TSMDS settings.

9 CONCLUSION

We presented Collaborative Self-Supervised Learning (ColloSSL), a new method to leverage unlabeled inertial data collected from multiple body-worn devices to learn a good representation of the data. In doing so, we exploited an important characteristic of the TSMDS setting that the time-aligned data from different devices can be considered as natural transformations of each other. Based on this observation, we presented a contrastive learning pipeline which intelligently gathers positive and negative samples from multiple devices, and contrasts them against a sample from the anchor device to generate a supervisory signal from unlabeled data.

Our key findings are that ColloSSL outperforms both fully-supervised and semi-supervised learning techniques in majority of the experiment settings. Secondly, ColloSSL is data-efficient and can outperform the fully-supervised baselines using one-tenth of the labeled data in most settings. We also showed that ColloSSL can learn well-separable features from the data, and threw light on how it makes its predictions by visualizing saliency maps. Even though our focus in this paper was on the task of human-activity recognition with inertial sensors, our idea of Collaborative Self-Supervised Learning and our proposed framework are general, and they can be applied to other sensing modalities and applications in future work.

REFERENCES

[1] P Aggarwal, Z Syed, X Niu, and N El-Sheimy. 2008. A standard testing and calibration procedure for low cost MEMS inertial sensors and units. *The Journal of Navigation* 61, 2 (2008), 323–336.

[2] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, Jorge Luis Reyes-Ortiz, et al. 2013. A public domain dataset for human activity recognition using smartphones.. In *Esann*, Vol. 3. 3.

[3] Philip Bachman, R Devon Hjelm, and William Buchwalter. 2019. Learning representations by maximizing mutual information across views. *arXiv preprint arXiv:1906.00910* (2019).

[4] Pierre Baldi. 2012. Autoencoders, Unsupervised Learning, and Deep Architectures. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning (Proceedings of Machine Learning Research, Vol. 27)*, Isabelle Guyon, Gideon Dror, Vincent Lemaire, Graham Taylor, and Daniel Silver (Eds.). PMLR, Bellevue, Washington, USA, 37–49. http://proceedings.mlr.press/v27/baldi12a.html

[5] Andreas Bulling, Ulf Blanke, and Bernt Schiele. 2014. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys (CSUR)* 46, 3 (2014), 1–33.

[6] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. 2021. Emerging Properties in Self-Supervised Vision Transformers. arXiv:2104.14294 [cs.CV]

[7] Youngjae Chang, Akhil Mathur, Anton Isopoussu, Junehwa Song, and Fahim Kawsar. 2020. A systematic study of unsupervised domain adaptation for robust human-activity recognition. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 1 (2020), 1–30.

[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*. PMLR, 1597–1607.

[9] Wenqiang Chen, Shupei Lin, Elizabeth Thompson, and John Stankovic. 2021. SenseCollect: We Need Efficient Ways to Collect On-body Sensor-based Human Activity Data! *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol* 1, 1 (2021).

[10] Xinlei Chen and Kaiming He. 2021. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 15750–15758.

[11] Shohreh Deldari, Daniel V Smith, Hao Xue, and Flora D Salim. 2021. Time Series Change Point Detection with Self-Supervised Contrastive Predictive Coding. In *Proceedings of the Web Conference 2021*. 3124–3135.

[12] Sanorita Dey, Nirupam Roy, Wenyuan Xu, Romit Roy Choudhury, and Srihari Nelakuditi. 2014. AccelPrint: Imperfections of Accelerometers Make Smartphones Trackable.. In *NDSS*. Citeseer.

[13] Davide Figo, Pedro C Diniz, Diogo R Ferreira, and Joao MP Cardoso. 2010. Preprocessing techniques for context recognition from accelerometer data. *Personal and Ubiquitous Computing* 14, 7 (2010), 645–662.

[14] Iuri Frosio, Federico Pedersini, and N Alberto Borghese. 2008. Autocalibration of MEMS accelerometers. *IEEE Transactions on Instrumentation and Measurement* 58, 6 (2008), 2034–2041.

[15] Andreas Grammenos, Cecilia Mascolo, and Jon Crowcroft. 2018. You Are Sensing, but Are You Biased?: A User Unaided Sensor Calibration Approach for Mobile Sensing. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 1, Article 11 (March 2018), 26 pages. https://doi.org/10.1145/3191743

[16] Arthur Gretton, Karsten Borgwardt, Malte J Rasch, Bernhard Scholkopf, and Alexander J Smola. 2008. A kernel method for the two-sample problem. *arXiv preprint arXiv:0805.2368* (2008).

[17] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. 2020. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733* (2020).

[18] Yu Guan and Thomas Plötz. 2017. Ensembles of Deep LSTM Learners for Activity Recognition using Wearables. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 2 (Jun 2017), 1–28. https://doi.org/10.1145/3090076

[19] Nils Y Hammerla, Shane Halloran, and Thomas Plötz. 2016. Deep, convolutional, and recurrent models for human activity recognition using wearables. *arXiv preprint arXiv:1604.08880* (2016).

[20] Nils Y Hammerla, Reuben Kirkham, Peter Andras, and Thomas Ploetz. 2013. On preserving statistical characteristics of accelerometry data using their empirical cumulative distribution. In *Proceedings of the 2013 international symposium on wearable computers*. 65–68.

[21] Harish Haresamudram, Apoorva Beedu, Varun Agrawal, Patrick L Grady, Irfan Essa, Judy Hoffman, and Thomas Plötz. 2020. Masked reconstruction based self-supervision for human activity recognition. In *Proceedings of the 2020 International Symposium on Wearable Computers*. 45–49.

[22] Harish Haresamudram, Irfan Essa, and Thomas Plötz. 2021. Contrastive predictive coding for human activity recognition. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5, 2 (2021), 1–26.

[23] Adam W. Harley, Shrinidhi K. Lakshmikanth, Fangyu Li, Xian Zhou, Hsiao-Yu Fish Tung, and Katerina Fragkiadaki. 2020. Learning from Unlabelled Videos Using Contrastive Predictive Neural 3D Mapping. arXiv:1906.03764 [cs.CV]

[24] Khalid Hasan, Kamanashis Biswas, Khandakar Ahmed, Nazmus S Nafi, and Md Saiful Islam. 2019. A comprehensive review of wireless body area network. *Journal of Network and Computer Applications* 143 (2019), 178–198.

[25] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum Contrast for Unsupervised Visual Representation Learning. arXiv:1911.05722 [cs.CV]

[26] Longlong Jing and Yingli Tian. 2020. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence* (2020).

[27] Seungwoo Kang, Youngki Lee, Chulhong Min, Younghyun Ju, Taiwoo Park, Jinwon Lee, Yunseok Rhee, and Junehwa Song. 2010. Orchestrator: An active resource orchestration framework for mobile context monitoring in sensor-rich mobile environments. In *2010 ieee international conference on pervasive computing and communications (percom)*. IEEE, 135–144.

[28] Matthew Keally, Gang Zhou, Guoliang Xing, Jianxin Wu, and Andrew Pyles. 2011. Pbn: towards practical activity recognition using smartphone-based body sensor networks. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*. 246–259.

[29] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2021. Supervised Contrastive Learning. arXiv:2004.11362 [cs.LG]

[30] Kai Kunze and Paul Lukowicz. 2008. Dealing with Sensor Displacement in Motion-Based Onbody Activity Recognition Systems. In *Proceedings of the 10th International Conference on Ubiquitous Computing* (Seoul, Korea) *(UbiComp '08)*. Association for Computing Machinery, New York, NY, USA, 20–29. https://doi.org/10.1145/1409635.1409639

[31] Song-Mi Lee, Sang Min Yoon, and Heeryon Cho. 2017. Human activity recognition from accelerometer data using Convolutional Neural Network. In *2017 ieee international conference on big data and smart computing (bigcomp)*. IEEE, 131–134.

[32] Jonathan Liono, A Kai Qin, and Flora D Salim. 2016. Optimal time window for temporal segmentation of sensor streams in multi-activity recognition. In *Proceedings of the 13th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*. 10–19.

[33] Akhil Mathur, Tianlin Zhang, Sourav Bhattacharya, Petar Velickovic, Leonid Joffe, Nicholas D Lane, Fahim Kawsar, and Pietro Lió. 2018. Using deep data augmentation training to address software and hardware heterogeneities in wearable and smartphone sensing devices. In *2018 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. IEEE, 200–211.

[34] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2019. A survey on bias and fairness in machine learning. *arXiv preprint arXiv:1908.09635* (2019).

[35] Daniela Micucci, Marco Mobilio, and Paolo Napoletano. 2017. Unimib shar: A dataset for human activity recognition using acceleration data from smartphones. *Applied Sciences* 7, 10 (2017), 1101.

[36] Chulhong Min, Alessandro Montanari, Akhil Mathur, and Fahim Kawsar. 2019. A closer look at quality-aware runtime assessment of sensing models in multi-device environments. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*. 271–284.

[37] Francisco Javier Ordóñez and Daniel Roggen. 2016. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors* 16, 1 (2016), 115.

[38] Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. 2019. Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779* (2019).

[39] Liangying Peng, Ling Chen, Zhenan Ye, and Yi Zhang. 2018. Aroma: A deep multi-task learning based simple and complex human activity recognition method using wearable sensors. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 2 (2018), 1–16.

[40] Thomas PlÖtz. 2021. Applying Machine Learning for Sensor Data Analysis in Interactive Systems: Common Pitfalls of Pragmatic Use and Ways to Avoid Them. *ACM Computing Surveys (CSUR)* 54, 6 (2021), 1–25.

[41] Thomas Plötz, Nils Y Hammerla, and Patrick L Olivier. 2011. Feature learning for activity recognition in ubiquitous computing. In *Twenty-second international joint conference on artificial intelligence*.

[42] Shashi Poddar, Vipan Kumar, and Amod Kumar. 2017. A comprehensive overview of inertial sensor calibration techniques. *Journal of Dynamic Systems, Measurement, and Control* 139, 1 (2017).

[43] Attila Reiss and Didier Stricker. 2012. Introducing a new benchmarked dataset for activity monitoring. In *2012 16th international symposium on wearable computers*. IEEE, 108–109.

[44] Daniel Roggen, Alberto Calatroni, Mirco Rossi, Thomas Holleczek, Kilian Förster, Gerhard Tröster, Paul Lukowicz, David Bannach, Gerald Pirkl, Alois Ferscha, et al. 2010. Collecting complex activity datasets in highly rich networked sensor environments. In *2010 Seventh international conference on networked sensing systems (INSS)*. IEEE, 233–240.

[45] Charissa Ann Ronao and Sung-Bae Cho. 2016. Human activity recognition with smartphone sensors using deep learning neural networks. *Expert systems with applications* 59 (2016), 235–244.

[46] Aaqib Saeed, Tanir Ozcelebi, and Johan Lukkien. 2019. Multi-task self-supervised learning for human activity detection. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 2 (2019), 1–30.

[47] Aaqib Saeed, Flora D Salim, Tanir Ozcelebi, and Johan Lukkien. 2020. Federated Self-Supervised Learning of Multisensor Representations for Embedded Intelligence. *IEEE Internet of Things Journal* 8, 2 (2020), 1030–1040.

[48] Aaqib Saeed, Victor Ungureanu, and Beat Gfeller. 2020. Sense and Learn: Self-Supervision for Omnipresent Sensors. *arXiv preprint arXiv:2009.13233* (2020).

[49] Bardia Safaei, Amir Mahdi Hosseini Monazzah, Milad Barzegar Bafroei, and Alireza Ejlali. 2017. Reliability side-effects in Internet of Things application layer protocols. In *2017 2nd International Conference on System Reliability and Safety (ICSRS)*. IEEE, 207–212.
[50] Pritam Sarkar and Ali Etemad. 2020. Self-supervised ecg representation learning for emotion recognition. *arXiv preprint arXiv:2002.03898* (2020).
[51] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. 2018. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 1134–1141.
[52] Haizhou Shi, Youcai Zhang, Zijin Shen, Siliang Tang, Yaqian Li, Yandong Guo, and Yueting Zhuang. 2021. Federated Self-Supervised Contrastive Learning via Ensemble Similarity Distillation. *arXiv preprint arXiv:2109.14611* (2021).
[53] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034* (2013).
[54] Maja Stikic, Kristof Van Laerhoven, and Bernt Schiele. 2008. Exploring semi-supervised and active learning for activity recognition. In *2008 12th IEEE International Symposium on Wearable Computers*. IEEE, 81–88.
[55] 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*. 127–140.
[56] Timo Sztyler and Heiner Stuckenschmidt. 2016. On-body localization of wearable devices: An investigation of position-aware activity recognition. In *2016 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 1–9.
[57] Chi Ian Tang, Ignacio Perez-Pozuelo, Dimitris Spathis, Soren Brage, Nick Wareham, and Cecilia Mascolo. 2021. SelfHAR: Improving Human Activity Recognition through Self-training with Unlabeled Data. *arXiv preprint arXiv:2102.06073* (2021).
[58] Chi Ian Tang, Ignacio Perez-Pozuelo, Dimitris Spathis, and Cecilia Mascolo. 2020. Exploring Contrastive Learning in Human Activity Recognition for Healthcare. *arXiv preprint arXiv:2011.11542* (2020).
[59] Tatiana Tommasi, Novi Patricia, Barbara Caputo, and Tinne Tuytelaars. 2017. A deeper look at dataset bias. In *Domain adaptation in computer vision applications*. Springer, 37–55.
[60] Antonio Torralba and Alexei A Efros. 2011. Unbiased look at dataset bias. In *CVPR 2011*. IEEE, 1521–1528.
[61] Yonatan Vaizman, Nadir Weibel, and Gert Lanckriet. 2018. Context recognition in-the-wild: Unified model for multi-modal sensors and multi-label classification. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 4 (2018), 1–22.
[62] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).
[63] Jesper E Van Engelen and Holger H Hoos. 2020. A survey on semi-supervised learning. *Machine Learning* 109, 2 (2020), 373–440.
[64] Jian Bo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiao Li Li, and Shonali Krishnaswamy. 2015. Deep Convolutional Neural Networks on Multichannel Time Series for Human Activity Recognition. In *Proceedings of the 24th International Conference on Artificial Intelligence* (Buenos Aires, Argentina) *(IJCAI'15)*. AAAI Press, 3995–4001.
[65] 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*. 351–360.
[66] Shuochao Yao, Yiran Zhao, Shaohan Hu, and Tarek Abdelzaher. 2018. Qualitydeepsense: Quality-aware deep learning framework for internet of things applications with sensor-temporal attention. In *Proceedings of the 2nd International Workshop on Embedded and Mobile Deep Learning*. 42–47.
[67] Piero Zappi, Clemens Lombriser, Thomas Stiefmeier, Elisabetta Farella, Daniel Roggen, Luca Benini, and Gerhard Tröster. 2008. Activity recognition from on-body sensors: accuracy-power trade-off by dynamic sensor selection. In *European Conference on Wireless Sensor Networks*. Springer, 17–33.
[68] Xiaojin Jerry Zhu. 2005. *Semi-supervised learning literature survey*. Technical Report. University of Wisconsin-Madison Department of Computer Sciences.