Multi-task Self-Supervised Learning for Human Activity Detection

AAQIB SAEED, Eindhoven University of Technology, The Netherlands
TANIR OZCELEBI, Eindhoven University of Technology, The Netherlands
JOHAN LUKKIEN, Eindhoven University of Technology, The Netherlands

Deep learning methods are successfully used in applications pertaining to ubiquitous computing, pervasive intelligence, health, and well-being. Specifically, the area of human activity recognition (HAR) is primarily transformed by the convolutional and recurrent neural networks, thanks to their ability to learn semantic representations directly from raw input. However, in order to extract generalizable features massive amounts of well-curated data are required, which is a notoriously challenging task; hindered by privacy issues and annotation costs. Therefore, unsupervised representation learning (i.e., learning without manually labeling the instances) is of prime importance to leverage the vast amount of unlabeled data produced by smart devices. In this work, we propose a novel self-supervised technique for feature learning from sensory data that does not require access to any form of semantic labels, i.e., activity classes. We learn a multi-task temporal convolutional network to recognize transformations applied on an input signal. By exploiting these transformations, we demonstrate that simple auxiliary tasks of the binary classification result in a strong supervisory signal for extracting useful features for the down-stream task. We extensively evaluate the proposed approach on several publicly available datasets for smartphone-based HAR in unsupervised, semi-supervised and transfer learning settings. Our method achieves performance levels superior to or comparable with fully-supervised networks trained directly with activity labels, and it performs significantly better than unsupervised learning through autoencoders. Notably, for the semi-supervised case, the self-supervised features substantially boost the detection rate by attaining a kappa score between $0.7 - 0.8$ with only 10 labeled examples per class. We get similar impressive performance even if the features are transferred from a different data source. Self-supervision drastically reduces the requirement of labeled activity data, effectively narrowing the gap between supervised and unsupervised techniques for learning meaningful representations. While this paper focuses on HAR as the application domain, the proposed approach is general and could be applied to a wide variety of problems in other areas.

CCS Concepts: • **Human-centered computing → Ubiquitous and mobile computing systems and tools**; • **Computing methodologies → Machine learning**;

Additional Key Words and Phrases: Self-supervised learning, multi-task learning, representation learning, semi-supervised learning, transfer learning, temporal convolutional neural networks, human activity recognition, deep learning

ACM Reference Format:
Aaqib Saeed, Tanir Ozcelebi, and Johan Lukkien. 2019. Multi-task Self-Supervised Learning for Human Activity Detection. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 3, 2, Article 61 (June 2019), 30 pages. https://doi.org/10.1145/3328932

1 INTRODUCTION

Over the last years, deep neural networks have been widely adopted for time-series and sensory data processing; achieving impressive performance in several application areas pertaining to pervasive sensing, ubiquitous

Authors' addresses: Aaqib Saeed, Eindhoven University of Technology, Eindhoven, The Netherlands, a.saeed@tue.nl; Tanir Ozcelebi, Eindhoven University of Technology, Eindhoven, The Netherlands, t.ozcelebi@tue.nl; Johan Lukkien, Eindhoven University of Technology, Eindhoven, The Netherlands, j.j.lukkien@tue.nl.

2474-9567/2019/6-ART61 $15.00
https://doi.org/10.1145/3328932

computing, industries, health and well-being [17, 21, 38, 56, 61, 73]. In particular, for smartphone-based human activity recognition (HAR), 1D convolutional and recurrent neural networks trained on raw labeled signals significantly improve the detection rate over traditional methods [20, 44, 68, 72, 73]. Despite the recent advances in the field of HAR, learning representations from a massive amount of unlabeled data still presents a significant challenge. Obtaining large, well-curated activity recognition datasets is problematic due to a number of issues. First, smartphone data are privacy sensitive, which makes it hard to collect sufficient amounts of user-activity instances in a real-life setting. Second, the annotation cost and the time it takes to generate a large volume of labeled instances are prohibitive. Finally, the diversity of devices, types of embedded sensors, variations in phone-usage, and different environments are further roadblocks in producing massive human-labeled data. To sum up, such expensive and hard to scale process of gathering labeled data generated by smart devices makes it very difficult to apply supervised learning in this domain directly.

In light of these challenges, we pose the question *whether it is possible to learn semantic representations in an unsupervised way to circumvent the manual annotation of the sensor data with strong labels, e.g., activity classes. In particular, the goal is to extract features that are on par with those learned with fully-supervised* [1] *methods*. There is an emerging paradigm for feature learning called *self-supervised learning* that defines auxiliary (also known as pretext or surrogate) tasks to solve, where labels are readily extractable from the data without any human intervention, i.e., self-supervised. The availability of strong supervisory signals from the surrogate tasks enables us to leverage objective functions as utilized in a standard supervised learning setting [14]. For instance, the vision community proposed a considerable number of self-supervised tasks for advancing representation learning[2] from static images, videos, and audio (see Section 5). Most prominent among them are: colorization of grayscale images [30, 74], predicting image rotations [18], solving jigsaw puzzles [47], predicting the direction of video playback [71], temporal order verification [42], odd sequence detection [15], audio-visual correspondence [3, 52], and curiosity-driven agents [54]. The presented methodology for sensor representation learning takes inspiration from these methods and takes leverage of signal transformations to extract highly generalizable features for the down-stream[3] task, i.e., HAR.

Our work is motivated by the success of jointly learning to solve multiple self-supervised tasks [10, 14] and we propose to learn accelerometer representations (i.e., features) by training a temporal convolutional neural network (CNN) to recognize the transformations applied to the raw input signal. Particularly, we utilize a set of signal transformations [7, 67] that are applied on each input signal in the datasets, which are then fed into the convolutional network along with the original data for learning to differentiate among them. In this simple formulation, a group of binary classification tasks (i.e., to recognize whether a transformation such as *permutation*, *scaling*, and *channel shuffling* was applied on the original signal or not) act as surrogate tasks to provide a rich supervisory signal to the model. In order to extract highly generalizable features for the end-task of interest, it is essential to utilize transformations that exploit versatile invariances of the temporal data (further details are provided in Section 3). To this end, we utilize eight transformations to train a multi-task network for simultaneously recognizing each of them. The visual illustration of the proposed approach is given in Figure 1. In the pre-training phase, the network consisting of a common trunk with a separate head for each task is trained on self-supervised data, and in the second step, the features learned by the shared layers are utilized by the HAR model. Importantly, we want to emphasize that in order for the convolutional network to recognize the transformations, it must learn to understand the core signal characteristics through acquiring knowledge of underlying differences in the accelerometer signals for various activity categories. We support this claim through an extensive evaluation of our method on six publicly available datasets in unsupervised, semi-supervised and

[1] The fully-supervised network is the standard deep model that is trained in an end-to-end fashion directly with activity labels without any pre-training.
[2] also known as feature learning
[3] or an end-task

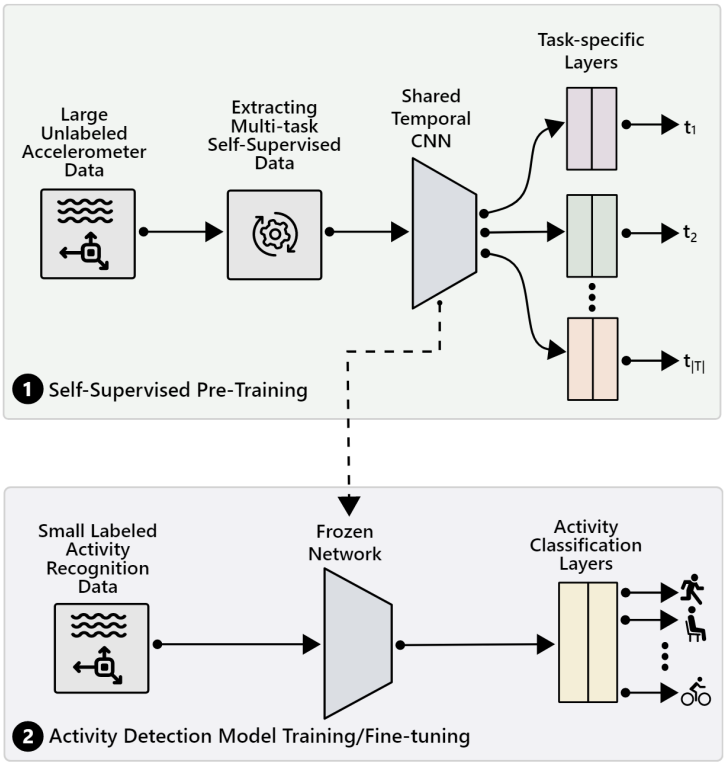


Fig. 1. Illustration of the proposed multi-task self-supervised approach for feature learning. We train a temporal convolutional network for transformation recognition as a pretext task as shown in Step 1. The learned features are utilized by (or transferred to) the activity recognition model (Step 2) for improved detection rate with a small labeled dataset.

transfer learning settings, where it achieves noticeable improvements in all the cases while not requiring manually labeled data for feature learning.

The main contributions of this paper are:

- We propose to utilize self-supervision from large unlabeled data for human activity recognition.
- We design a signal transformation recognition problem as a surrogate task for annotation-free supervision, which provides a strong training signal to the temporal convolutional network for learning generalizable features.
- We demonstrate through extensive evaluation that the self-supervised features perform significantly better in the semi-supervised and transfer learning settings on several publicly available datasets. Moreover, we show that these features achieve performance that is superior to or comparable with the features learned via the fully-supervised approach (i.e., trained directly with activity labels).
- We illustrate with SVCCA [57], saliency mapping [63], and t-SNE [39] visualizations that the features extracted via self-supervision are very similar to those learned by the fully-supervised network.
- Our method substantially reduces the labeled data requirement, effectively narrowing the gap between unsupervised and supervised representation learning.

The paper is organized as follows. Section 2 provides an overview of related paradigms and methodologies as background information. Section 3 introduces the proposed self-supervised representation learning framework for HAR. Section 4 presents an evaluation of our framework on publicly available datasets. Section 5 gives an overview of the related work. Finally, Section 6 concludes the paper and lists future directions for research.

2 PRELIMINARIES

In this section, we provide a brief overview of multiple learning paradigms, including multi-task, transfer, semi-supervised and importantly, representation learning. These either benefit or serve as fundamental building blocks of our self-supervised framework for representation extraction and robust HAR under various settings.

2.1 Representation Learning

Representation (feature) learning is concerned with automatically extracting useful information from the data that can be effectively used for an impending machine learning problem such as classification. In the past, most of the efforts were spent on developing (and manually engineering) feature extraction methods based on domain expertise to incorporate prior knowledge in the learning process. However, these methods are relatively limited as they rely on human creativity to come up with novel features and lack the power to capture underlying explanatory factors in the milieu of low-level sensory input. To overcome these limitations and to automate the discovery of disentangled features, neural networks based approaches have been widely utilized, such as autoencoders and their variants [6]. Deep neural networks are composed of multiple (parameterized) non-linear transformations that are trained through a supervised or unsupervised objective function with the aim of yielding useful representations. These techniques have achieved indisputable empirical success across a broad spectrum of problems [4, 21, 28, 36, 43, 56, 65, 66] thanks to the increasing dataset sizes and computing power availability. Nevertheless, representation learning still stands as a fundamental problem in machine intelligence and is an active area of research (see [8] for a detailed survey).

2.2 Multi-task Learning

The goal of multi-task learning (MTL) is to enhance the learning efficiency and accuracy through simultaneously optimizing multiple objectives based on shared representations and exploiting relations among the tasks [10]. It is widely utilized in several application domains within machine learning such as natural language processing [22], computer vision [25], audio sensing [17], and well-being [61]. In this learning setting, T supervised tasks, each with a dataset $D^t = \{x_i^t, y_i^t\}_{i=1}^m$ and a separate cost function are made available. The multi-objective loss is then generally created through a weighted linear sum of the individual tasks' losses as:

$$\mathcal{L}_{Aggregated} = \sum_{t \in T} \psi_t \mathcal{L}_t \tag{1}$$

where ψ_t is the task weight and $\mathcal{L}_t$ is a task-specific loss function. It is important to note that, MTL itself does not impose any restriction on the loss type of an individual task. Therefore, unsupervised and supervised tasks or tasks having different cost functions can be conveniently combined for learning representations.

2.3 Transfer Learning

Transfer learning aims to develop methods for preserving and leveraging previously acquired knowledge to accelerate the learning of novel tasks. In recent years, it has shown remarkable improvement in performance on several very challenging problems, especially in areas, where little-labeled data are available such as in natural language understanding, object recognition, and activity recognition [23, 44, 62]. In this paradigm, the goal is to transfer (or reuse) the learned knowledge from a source domain $\mathcal{D}_{SRC}$ to a target domain $\mathcal{D}_{TRG}$. More precisely, consider domains $\mathcal{D}_{SRC}$ and $\mathcal{D}_{TRG}$ with learning tasks t_{SRC} and t_{TRG}, respectively. The goal is to help improve

the learning of a predictive function $f(.)$ in t_{TRG} using the knowledge extracted from $\mathcal{D}_{SCR}$ and t_{SRC}, where $\mathcal{D}_{SRC} \neq \mathcal{D}_{TRG}$, and/or $t_{SRC} \neq t_{TRG}$, meaning that domains or tasks may be different. This learning formulation enables to develop a high-quality model under different knowledge transfer settings (such as features, instances, weights) from existing labeled data of some related task or domain. For a detailed review of transfer learning, we refer an interested reader to [53].

2.4 Semi-supervised Learning

Semi-supervised learning provides a compelling framework for leveraging unlabeled data in cases when labeled data collection is expensive. It has been repeatedly shown that given enough computational power and supervised data; deep neural networks can achieve human-level performance on a wide variety of problems [31]. However, the curation of large-scale datasets is very costly and time-consuming as it either requires crowdsourcing or domain expertise such as in the case of medical imaging. Likewise, for several practical problems, it is simply not possible to create a large enough labeled dataset (e.g., due to privacy issues) to learn a model of reasonable accuracy. Semi-supervised learning algorithms offer a compelling alternative to fully-supervised methods for jointly learning from few labeled and a large number of unlabeled instances. More specifically, given a labeled training set of input-output pairs $(X, Y) \in D_L$ and unlabeled instance set, $X \in D_U$, the broad aim is to produce a predictive function $f_\theta(X)$ making use of not only D_L but also the underlying structure in D_U, where θ represents the learnable parameters of the model. For a concise review and realistic evaluation of various deep learning based semi-supervised techniques, see [50].

2.5 Towards Self-supervision

Deep learning has been increasingly used for end-to-end HAR with far superior performance that can be achieved through traditional machine learning methods [20, 44, 56, 60, 72]. However, learning from very few labeled data, i.e. few-shot and semi-supervised learning is still an issue as large labeled datasets are required to train a model of sufficient quality. Similarly, the utilization of previously learned knowledge from related data (or task) to rapidly solve a comparable problem is not addressed very well by the existing methods (see Section 5 for more details). In this paper, we explore self-supervised feature learning for HAR that effectively utilizes unlabeled data. The exciting field of self-supervision is concerned with extracting supervisory signals from data without requiring any human intervention. The evolution of feature extraction methods from hand-crafted features towards self-supervised representations is illustrated in Figure 2. The input to each of the illustrated approaches is raw data, which is not shown for the sake of brevity.

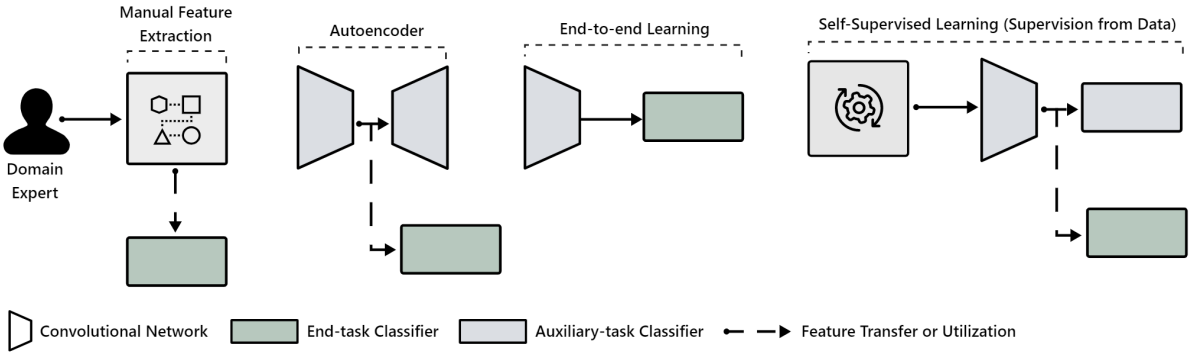


Fig. 2. Evolution of feature learning approaches from hand-crafted methods towards task discovery for self-supervision.

3 APPROACH

In this section, we present our self-supervised representation learning framework for HAR. First, we provide an overview of the methodology. Next, we discuss various learning tasks (i.e. transformation classification) and their benefits for generic features extraction from unlabeled data. Finally, we provide a detailed description of the network architecture, its implementation, and the optimization process.

3.1 Overview

The objective of our work is to learn general-purpose sensor representations based on a temporal convolutional network in an unsupervised manner. To achieve this goal, we introduce a self-supervised deep network named *Transformation Prediction Network* (TPN), which simultaneously learns to solve multiple (signal) transformation recognition tasks as shown in Figure 1. Specifically, the proposed multi-task TPN $M_\theta(.)$ is trained to produce estimates of the transformations applied to the raw input signal. We define a set of $|T|$ distinct transformations (or tasks) $T = \{J_t(.)\}_{t \in T}$, where $J_t(.)$ is a function that applies a particular signal alteration technique t to the temporal sequence $x \in \mathbb{R}^{(N,C)}$ to yield a transformed version of the signal $J_t(x)$. The network $M_\theta(.)$ that has a common trunk and individual head for each task, it takes an input sequence and produces a probability of the signal being a transformed version of the original, i.e. $P(J_t|x) = M_\theta(x)$. Note, that given a set of unlabeled signals (e.g. of accelerometer), we can automatically construct a self-supervised labeled dataset $D = \{\{(J_t(x_i), True), (x_i, False)\}_{t \in T}\}_{i=1}^{m}$. Hence, given this set of m training instances, the multi-task self-supervised training objective that a model must learn to solve is:

$$\min_{\theta} \sum_{t \in T} \psi_t \Big[-\frac{1}{m_t} \sum_{i=1}^{m_t} (y_i^t \log(M_\theta(x_i^t)) + (1 - y_i^t) \log(1 - M_\theta(x_i^t))) \Big] \tag{2}$$

where $M_\theta(x^t)$ is the predicted probability of x being a transformed version t and θ are the learnable parameters of the network. m_t represents the number of instances for a task (which can vary but are equal in our case) and ψ_t is the loss-weight of task t.

We emphasize that, although the network has a separate layer to differentiate between original and each of the T transformations it can be extended in a straight-forward manner to recognize multiple transformations applied to the same input signal or for multi-label classification. In the following subsection, we explain the types of signal transformations that are used in this work.

3.2 Self-supervised Tasks: *Signal Transformations*

The aforementioned formulation requires the signal transformations J to define a multi-task classification that enables the convolutional model to learn disentangled semantic representations useful for down-stream tasks, e.g. activity detection. We aimed for conceptually simple, yet diverse tasks to possibly cover several invariances that commonly arise in temporal data [7]. Intuitively, a diverse set of tasks should lead to a broad spectrum of features, which are more likely to span the feature-space domain needed for a general understanding of the signal's characteristics. In this work, we propose to utilize eight straight-forward signal transformations (i.e. $|T| = 8$) [7, 67] for the self-supervision of a network. More specifically, when transformations are applied on an input signal x, they result in eight variants of x. As mentioned earlier, the temporal convolutional model is then trained jointly on all the tasks' data to solve a problem of transformation recognition, which allows the model to extract high-level abstractions from the raw input sequence. The transformations utilized in this work are summarized below:

- **Noised:** Given sensor readings of a fixed length, a possible transformation is the addition of random noise (or jitter) in the original signal. Heterogeneity of device sensors, software, and other hardware can cause

variations (noisy samples) in the produced data. A model that is robust against noise will generalize better as it learns features that are invariant to minor corruption in the signal.

- **Scaled:** A transformation that changes the magnitude of the samples within a window through multiplying with a randomly selected scalar. A model capable of handling scaled signals produces better representations as it becomes invariant to amplitude and offset invariances.
- **Rotated:** Robustness against arbitrary rotations applied on the input signal can achieve sensor-placement (orientation) invariance. This transformation inverts the sample signs (without changing the associated class-label) as frequently happens if the sensor (or device) is, for example, held upside down.
- **Negated:** This simple transformation is an instance of both *scaled* (scaling by -1) and *rotated* transformations. It negates samples within a time window, resulting in a vertical flip or a mirror image of the input signal.
- **Horizontally Flipped:** This transformation reverses the samples along the time-dimension, resulting in a complete mirror image of an original signal as if it were evolved in the opposite time direction.
- **Permuted:** This transformation randomly perturbs the events within a temporal window through slicing and swapping different segments of the time-series to generate a new one, hence, facilitating the model to develop permutation invariance properties.
- **Time-Warped:** This transformation locally stretches or warps a time-series through a smooth distortion of time intervals between the values (also known as local scaling).
- **Channel-Shuffled:** For a multi-component signal such as a triaxial accelerometer, this transformation randomly shuffles the axial dimensions.

There are several benefits of utilizing transformations recognition as auxiliary tasks for feature extraction from unlabeled data.

Enabling the learning of generic representations: The primary motivation is that the above-defined pretext tasks enable the network to capture the core signal characteristics. More specifically, for the TPN to successfully recognize if the signal is transformed or not, it must learn to detect high-level semantics, sensor behavior under different device placements, time-shift of the events, varying amplitudes, and robustness against sensor noise, thus, contributing to solving the ultimate task of HAR.

Task diversification and elimination of low-level input artifacts: A clear advantage of using multiple self-supervised tasks as opposed to a single one is that it will lead to a more diverse set of features that are invariant to low-level artifacts of the signals. Had we chosen to utilize signal reconstruction, e.g. with autoencoders, this would learn to compress the input, but due to a weak supervisory signal (as compared to self-supervision), it may discover trivial features with no practical value for the activity recognition or any other task of interest. We compare our approach against other methods in section 4.3.

Transferring knowledge: Furthermore, with our approach, the unlabeled sensor data that are produced in huge quantity can be effectively utilized with no human intervention to pre-train a network that is suitable for semi-supervised and transfer learning settings. It is particularly of high value for training networks in a real-world setting, where very little or no supervision is available to learn a model of sufficient quality from scratch.

Other benefits: Our self-supervised method has numerous other benefits. It has an equivalent computational cost to supervised learning but with better convergence accuracy, making it a suitable candidate for continuous unsupervised representation learning in-the-wild. Moreover, our technique neither requires a sophisticated pre-processing (apart from z-normalization) nor needs a specialized architecture (which also requires labeled data) to exploit invariances. We will show in Section 4.3 through extensive evaluation that the self-supervised models learn useful representations and dramatically improve performance over other learning strategies. Despite

the simplicity of the proposed scheme, it allows utilizing data collected through a wide variety of devices from a diverse set of users.

3.3 Network Architecture and Implementation

We implement the TPN $M_\theta(.)$ as a multi-branch temporal convolutional neural network with a common trunk (shared layers) and a distinct head (private layers) for each task with a separate loss function. Hard parameter sharing is employed between all the task-specific layers to encourage strong weight utilization from the trunk. Figure 3 illustrates the TPN containing three 1D convolutional layers consisting of 32, 64, and 96 feature maps with kernel sizes of 24, 16 and 8 respectively, and having a stride of 1. Dropout is used after each of the layers with a rate of 0.1, and L2 regularization is applied with a rate of 0.0001. Global max pooling is used after the last convolution layer to aggregate high-level discriminative features. Moreover, each task-specific layer is comprised of a fully-connected layer of 256 hidden units followed by a sigmoidal output layer for binary classification. We use *ReLU* as non-linearity in all the layers (except the output) and train a network with Adam optimizer [26] for a maximum of 30 epochs with a learning rate of 0.0003, unless stated otherwise. Furthermore, the activity recognition model has a similar architecture to the TPN except for a fully-connected layer that consists of 1024 hidden units followed by a softmax output layer with units depending on the activity detection task under consideration. Additionally, during training of this model, we apply early-stopping, if the network fully converges on the training set to avoid overfitting.

The motivation for keeping the TPN architecture simple arises from the fact that we want to show the performance gain does not come from the number of parameters (or layers) or due to the utilization of other sophisticated techniques such as batch normalization but the improvement is due to self-supervised pre-training. Likewise, the choice of multi-task learning setting, where each task has an additional private layer manifests in letting the model push pretext task-specific features to the last layers and let the initial layers extract generic representations that are important for a wide variety of end-tasks. Moreover, our architectural specification allows for a straightforward extension to add other related tasks, if needed, such as input reconstruction. Although, we do not explore applying multiple transformations to the same sequence or train models for their recognition the network design is intrinsically capable of performing this multi-label classification task.

Our training process is summarized in Algorithm 1. For every instance, we first generate transformed versions of a signal for the self-supervised pre-training of the network. At each training iteration of the TPN model, we feed the data from all tasks simultaneously, and the overall loss is calculated as a weighted sum of the losses of different tasks. Once pre-training converges, we transfer the weights of convolutional layers from model M_θ to an activity recognition network C_θ for learning the final supervised task. Here, either all the transferred layers are kept frozen, or the last convolutional layer is fine-tuned depending on the learning paradigm. Figure 3 depicts this process graphically, where shaded convolutional layers represent frozen weights, while others are either trained from scratch or optimized further on the end-task. To avoid ambiguity, in the experiment section, we explicitly mention when the results are from a fully-supervised or self-supervised (including fine-tuned) network.

4 EVALUATION

In this section, we conduct an extensive evaluation of our approach on several publicly available datasets for human activity recognition (HAR) in order to determine the quality of learned representations, transferability of the features, and benefits of this in the low-data regime. The self-supervised tasks (i.e., transformation predictions) are utilized for learning rich sensor representations that are suitable for an end-task. We emphasize that achieving high performance on these surrogate tasks is **not** our focus.

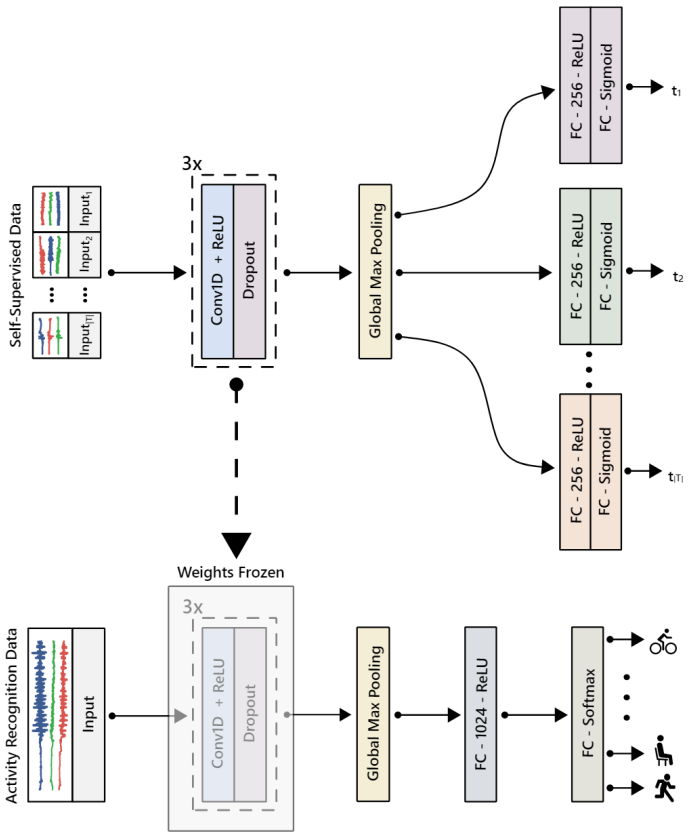


Fig. 3. Detailed architectural specification of transformation prediction and activity recognition networks. We propose a framework for self-supervised representation learning from unlabeled sensor data (such as an accelerometer). Various signal transformations are utilized to establish supervisory tasks, and the network is trained to differentiate between an original and transformed version of the input. The three blocks of *Conv + ReLU* and *Dropout* layers, which is followed by a *Global Max Pooling* are similar across both networks. However, the multi-task model has a separate head for each task. Likewise, the activity recognizer has an additional densely connected layer. The TPN is pre-trained on self-supervised data, and the learned weights are transferred (depicted by a dashed arrow) and kept frozen to the lower model, which is then trained to detect various activities.

4.1 Datasets

We consider six publicly available datasets to cover a wide variety of device types, data collection protocols, and activity recognition tasks performed with smartphones in different environments. Some important aspects of the data are summarized in Table 1. Below, we give brief descriptions of every dataset summarizing its key points.

4.1.1 HHAR. The Heterogeneity Human Activity Recognition (HHAR) dataset [64] contains signals from two sensors (accelerometer and gyroscope) of smartphones and smartwatches for 6 different activities, i.e. biking, sitting, standing, walking, stairs-up and stairs-down. The 9 participants executed a scripted set of activities for 5 minutes to get equal class distribution. The subjects had 8 smartphones in a tight pouch carried around their

Algorithm 1: Multi-task Self-Supervised Learning

Input: Unlabeled instance set D_U, labeled dataset D_L, task-specific weights ψ, numbers of epochs E_M and E_C
Output: Self-supervised network M, activity classification model C with n classes
initialize $(X, Y)_1, \ldots, (X, Y)_T$ to hold instance-label pairs for multiple tasks in T
initialize M with parameters θ_M and C with parameters θ_C
`// Labeled data generation for self-supervision`
for *each instance* x *in* D_U **do**
  for *each transformation* $t \in T$ **do**
    Insert $(x, False)$ to $(X, Y)_t$ and $(J_t(x), True)$ to $(X, Y)_t$
  end
end
for *each epoch* e_m *from* 1 *to* E_M **do**
  Randomly sample a mini-batch of m samples for all tasks $\{(X, Y)_1, \ldots, (X, Y)_T\}$
  Update θ_M by descending along its gradient

$$\nabla_{\theta_M}\left[\sum_{t\in T}\psi_t\left[-\frac{1}{m_t}\sum_{i=1}^{m_t}(y_i^t\log(M_\theta(x_i^t))+(1-y_i^t)\log(1-M_\theta(x_i^t)))+\beta\,\|\theta\|^2\right]\right]$$

end
Assign learned parameters from $\theta_M^{1\cdots L}$ to $\theta_C^{1\cdots L}$
Keep the transferred weights $\theta_C^{1\cdots L}$ of network C frozen
for *each epoch* e_c *from* 1 *to* E_C **do**
  Randomly sample a mini-batch of m labeled activity recognition samples D_L
  Update θ_C by descending along its gradient $\nabla_{\theta_C}\left[-\left(\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{n}\ y_{i,k}\ \log(C_\theta(x_i))\right)+\beta(\|\theta\|^2)\right]$
end
Gradient-based updates can use any standard gradient-based learning technique. We used Adam [26] in all our experiments.

Table 1. Summary of datasets used in our evaluation. These datasets are selected based on the diversity of participants, device types and activity classes. Further details on the pre-processing of each data source and the number of users utilized are discussed in Section 4.1.

Dataset	No. of users	No. of activity classes
HHAR	9	6
UniMiB	30	9
UCI HAR	30	6
MobiAct	67	11
WISDM	36	6
MotionSense	24	6

waist and 4 smartwatches, 2 worn on each arm. In total, they used 36 different smart devices of 13 models from 4 manufacturers to cover a broad range of devices for sampling rate heterogeneity analysis. The sampling rate of signals varied significantly across phones with values between 50-200Hz.

4.1.2 UniMiB. This dataset [41] contains triaxial accelerometer signals collected from a Samsung Galaxy Nexus smartphone at 50Hz. Thirty subjects participated in the data collection process forming a diverse sample of the population with different height, weight, age, and gender. The subject placed the device in her trouser's front left pocket for a partial duration and in the right pocket for the remainder of the experiment. We utilized the data of 9 activities of daily living (i.e., standing up from sitting, standing up from lying, walking, running, upstairs, jumping, downstairs, lying down from sitting, sitting) in this paper.

4.1.3 UCI HAR. The UCI HAR dataset [2] is obtained from a group of 30 volunteers with a waist-mounted Samsung Galaxy S2 smartphone. The accelerometer and gyroscope signals are collected at 50Hz when subjects performed the following six activities: standing, sitting, laying down, walking, downstairs and upstairs.

4.1.4 MobiAct. The MobiAct[4] dataset [11] contains signals from a smartphone's inertial sensors (accelerometer, gyroscope, and orientation) for 11 different activities of daily living and 4 types of falls. It is collected with a Samsung Galaxy S3 smartphone from 66 participants of different gender, age group, and weight through more than 3200 trials. The device is placed in a trouser's pocket freely selected by the subject in any random orientation to capture everyday usage of the phone. We used the data from 61 participants who have data samples for any of the following 11 activities: sitting, walking, jogging, jumping, stairs up, stairs down, stand to sit, sitting on a chair, sit to stand, car step-in, and car step-out.

4.1.5 WISDM. The dataset from the Wireless Sensor and Data Mining (WISDM) project [29] was collected in a controlled study from 29 volunteers, who carried the cell phone in their pockets. The data were recorded for 6 different activities (i.e., sit, stand, walk, jog, ascend stairs, descend stairs) via an app developed for an Android phone. The accelerometer signal was acquired every 50ms (sampling rate of 20Hz). We use the data of all the users available in the raw data file with user ids ranging from 1 to 36.

4.1.6 MotionSense. The MotionSense dataset [40] comprises an accelerometer, gyroscope, and altitude data from 24 participants of varying age, gender, weight, and height. It was collected using an iPhone6s, which is kept in the user's front pocket. The subjects performed 6 different activities (i.e., walking, jogging, downstairs, upstairs, sitting, and standing.) in 15 trials under similar environments and conditions. The study aimed to infer physical and demographics attributes from time-series data in addition to the detection of activities.

4.2 Data Preparation and Assessment Strategy

We applied minimal pre-processing on the accelerometer signals as deep neural networks are very good at learning abstract representations directly from raw data [31]. We segmented the signals into fixed size windows that have 400 samples with 50% overlap, for all the datasets under consideration. The appropriate window size is a task-specific parameter and could be tuned or chosen based on prior knowledge for improved performance. Here, we utilize the same window size based on earlier exploration across datasets and to keep experimental evaluation impartial towards the effect of this hyper-parameter. Next, we divide each dataset into training and test sets through randomly selecting $20 - 30\%$ of the users for testing and the rest for training and validation; depending on the dataset size. We used the ceiling function to select number of users, e.g. from HHAR dataset 3 users are used for evaluation out of 9. The training set users' data are further divided into 80% for training the network and 20% for validation and hyper-parameter tuning. Importantly, we also evaluate our models through user-split based 5-folds cross-validation, wherever it is appropriate. Finally, we normalize the data by applying z-normalization with summary statistics calculated from the training set. We generate self-supervised data from an unlabeled training set that is produced as a result of the processing as mentioned earlier. We utilize the data generation procedure as explained earlier in Section 3.3.

[4] second release

Furthermore, due to the large size of the *HHAR* dataset and in order to reduce computational load, we randomly sample 4000 instances from each users' data to produce transformed signals. Likewise, in the case of *UniMiB* because of its relatively small size, we generate 5 times more transformed instances. We evaluate the performance with Cohen's kappa, a weighted version of precision, recall and f-score metrics to be robust against inherent imbalanced nature of the datasets. It is important to highlight that, *we use a network architecture with the same configuration across the datasets to evaluate models' performance in order to highlight improvement is indeed due to self-supervision and not due to architectural modifications.*

4.3 Results

4.3.1 Quantifying the Quality of Learned Feature Hierarchies. We first evaluate our approach to determine the quality of learned representations versus the model depth (i.e., the layer number from which the features come). This analysis helps in understanding *whether the features coming from different layers vary in quality concerning their performance on an end-task and if so, which layer should be utilized for this purpose.* To this end, we first pre-train our TPN in a self-supervised manner and learn classifiers on top of *ConvA*, *ConvB*, and *ConvC* layers independently, for several activity recognition datasets. These classifiers (see Figure 3) are trained in a supervised way while keeping the learned features fixed during the optimization process. Figure 4 provides kappa values on test sets averaged across 10-independent runs to be robust against differences in weight initializations of the classifiers. We observe that for a majority of the datasets the model performance improves with increasing depth apart from *HHAR*, where features from *ConvB* layer results in improved detection rate with a kappa of 0.774 compared to 0.679 of *ConvC*. It may be because the representation of the last layer starts to become too specific on the transformation prediction task or it may also be because we did not utilize the entire dataset for the self-supervision. To be consistent, in the subsequent experiments we used features from the last convolutional layer for all the considered datasets. For a new task or recognition problem, we recommend performing a similar analysis to identify layer/block of the network that gives optimal results on the particular dataset.

4.3.2 Comparison against Fully-Supervised and Unsupervised Approaches. In this subsection, we assess our self-supervised representations learned with TPN against other unsupervised and fully-supervised techniques for feature learning. Table 2 summarizes the results with respect to four evaluation metrics (namely, precision, recall, f-score, and kappa) for 10-independent runs on the six datasets described earlier. For the *Random Init.* entries, we keep the convolutional network layers frozen during optimization and train only a classifier in a supervised manner. Likewise, for an *Autoencoder*, we keep the network architecture the same and pre-train it in an unsupervised way. Afterward, the weights of the encoder are kept frozen, and a classifier is trained on top as usual. The *Self-Supervised* entries show the result of the convolutional network pre-trained with our proposed method, where a classifier is trained on top of the frozen network in a supervised fashion. Furthermore, *Self-Supervised (FT)* entries highlight the performance of the network trained with self-supervision but the last convolution layer, i.e. *ConvC* is fine-tuned along with a classifier during training on the activity recognition task. Training an activity classification model on top of randomly initialized convolutional layers poorly performs as expected, which is evidence that the performance improvement is not only because of the activity classifier. These results are followed by a widely used unsupervised learning method, i.e. an autoencoder. The self-supervised technique outperforms existing methods and achieves results that are on par with the fully-supervised model. It is important to note that, for our proposed technique, only the classifier layers are randomly initialized and trained with activity specific labels (the rest is transferred from the self-supervised network). We also observe that fine-tuning the last convolutional layer further improves the classification performance of the down-stream tasks on several datasets such as *UniMiB*, *HHAR*, *MobiAct*, and *UCI HAR*. The results show that TPN can learn highly generalizable representations, thus reducing the performance gap of feature learning with the (end-to-end) supervised case. For a more rigorous evaluation, we also performed 5-folds (user split based) cross-validation for

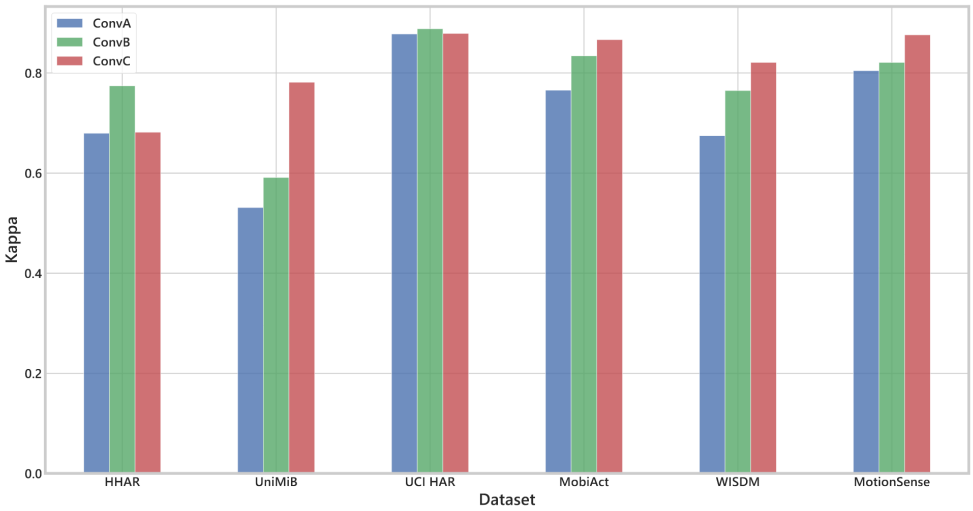


Fig. 4. Evaluation of activity classification performance using the features learned based on self-supervision (per layer). We train an activity classifier on-top of each of the temporal convolution blocks (*ConvA*, *ConvB*, and *ConvC*) that are pre-trained with self-supervision. The reported results are averaged over 10 independent runs (i.e., training an activity classifier from scratch). *ConvA*, *ConvB*, and *ConvC* have 32, 64, and 96 feature maps, respectively.

every method on all the datasets. The results are provided in Table 4 of the appendix, which also shows that the self-supervised method reduces the performance gap with the supervised setting.

4.3.3 Assessment of Individual Self-Supervised Tasks in Contrast with Multiple Tasks. In Figure 5, we show comparative performance analysis of single self-supervised tasks with each other and importantly with a multi-task setting. This assessment helps us in understanding *whether self-supervised features extracted via jointly learning to solve multiple tasks are any better (for activity classification) than independently solving individual tasks* and *whether multi-task learning helps in learning more useful sensor semantics*. To achieve this, we pre-train a TPN on each of the self-supervised tasks and transfer the weights for learning an activity recognition classifier. We observe in all the cases that learning representations via solving multiple tasks lead to far better performance on the end-task. This further highlights that the features learned through various self-supervised tasks have different strengths and weaknesses. Therefore, merging multiple tasks results in an improvement in learning a diverse set of features. However, we notice that some tasks (such as *Channel Shuffled*, *Permuted*, and *Rotated*) consistently performed better compared to others across datasets; achieving a kappa score above 0.60 as evaluated on different activity recognition problems. It highlights an important point that there may exist a group of tasks, which are reasonably sufficient to achieve a model of good quality. Furthermore, in Figure 11 of the appendix, we plot the kappa score achieved by a multi-task TPN on transformation recognition tasks as a function of the number of training epochs. This analysis highlights that task complexity varies greatly from one dataset to another and may help with the identification of trivial auxiliary tasks that may lead to non-generalizable features.

In addition to activity classification, for any learning task involving time-series sensor data (e.g., as encountered in a various Internet of Things applications), we recommend extracting features through first solving individual tasks and later focusing on the multi-task scenario; discarding low performing tasks or assigning low-weights to the loss functions of the respective tasks. Another approach could be to auto-tune the task-loss weight by taking homoscedastic uncertainty of each task into account [25].

Table 2. Task Generalization: Evaluating self-supervised representations for activity recognition. We compare the proposed self-supervised method for representation learning with fully-supervised and unsupervised approaches. We use the same architecture across all the experiments. The self-supervised TPN is trained to recognize transformations applied on the input signal while the activity classifier is trained on top of these learned features where *Self-Supervised (FT)* entry provides results when the last convolution layer is fine-tuned. The *Random Init.* entries present results when the convolution layers are randomly initialized and kept frozen during the training of the classifier. The results reported are averaged over 10 independent runs to be robust against variations in the weight initialization and the optimization process.

(a) HHAR

	P	**R**	**F**	**K**
Random Init.	0.3882±0.0557	0.3101±0.0409	0.2141±0.0404	0.1742±0.0488
Supervised	0.7624±0.0312	0.7353±0.0308	0.7276±0.0297	0.6816±0.0371
Autoencoder	0.7317±0.0451	0.6657±0.0663	0.6585±0.0724	0.5994±0.0784
Self-Supervised	0.7985±0.0155	0.777±0.0199	0.7666±0.0234	0.731±0.0243
Self-Supervised (FT)	0.8218±0.0256	0.797±0.0211	0.7862±0.0187	**0.7555±0.025**

(b) UniMiB

	P	**R**	**F**	**K**
Random Init.	0.4256±0.0468	0.3546±0.037	0.2775±0.0491	0.2243±0.0474
Supervised	0.8276±0.0148	0.8096±0.0266	0.8097±0.0248	0.7815±0.0299
Autoencoder	0.5922±0.0191	0.5557±0.0232	0.5376±0.0339	0.4824±0.0275
Self-Supervised	0.8133±0.0077	0.7954±0.014	0.7929±0.016	0.7642±0.0162
Self-Supervised (FT)	0.8506±0.007	0.8432±0.0049	0.8425±0.0054	**0.8197±0.005**

(c) UCI HAR

	P	**R**	**F**	**K**
Random Init.	0.6189±0.0648	0.4392±0.0692	0.3713±0.0952	0.3133±0.0866
Supervised	0.9059±0.0133	0.8998±0.0139	0.8981±0.0148	0.8789±0.0168
Autoencoder	0.8314±0.0590	0.7877±0.1112	0.7772±0.1306	0.7425±0.1359
Self-Supervised	0.9100±0.0081	0.9011±0.0139	0.8987±0.0155	**0.8803±0.0169**
Self-Supervised (FT)	0.9057±0.0121	0.897±0.0185	0.8946±0.019	0.8754±0.0222

(d) MobiAct

	P	**R**	**F**	**K**
Random Init.	0.4749±0.1528	0.3452±0.1128	0.2813±0.0982	0.1915±0.1017
Supervised	0.908±0.0066	0.895±0.0167	0.8975±0.0133	0.8665±0.0202
Autoencoder	0.7493±0.0328	0.7581±0.0354	0.7293±0.0452	0.6772±0.0517
Self-Supervised	0.9095±0.0035	0.9059±0.0059	0.906±0.0053	0.8795±0.0073
Self-Supervised (FT)	0.9194±0.0057	0.9102±0.0114	0.9117±0.0093	**0.8855±0.014**

(e) WISDM

	P	**R**	**F**	**K**
Random Init.	0.5942±0.0599	0.3543±0.077	0.358±0.0837	0.2224±0.0656
Supervised	0.9024±0.0076	0.8657±0.0206	0.8764±0.0168	0.8211±0.0258
Autoencoder	0.6561±0.2775	0.6631±0.1623	0.6358±0.2355	0.5106±0.288
Self-Supervised	0.8894±0.0096	0.8484±0.0269	0.8593±0.0225	0.7986±0.0334
Self-Supervised (FT)	0.8999±0.0111	0.8568±0.0375	0.8686±0.0314	0.8106±0.0466

(f) MotionSense

	P	**R**	**F**	**K**
Random Init.	0.5999±0.0956	0.5029±0.0931	0.4681±0.1105	0.376±0.1176
Supervised	0.9164±0.0053	0.8993±0.0091	0.9027±0.0085	0.8763±0.011
Autoencoder	0.8255±0.0132	0.8116±0.0195	0.8109±0.0169	0.7659±0.0226
Self-Supervised	0.8979±0.0073	0.8856±0.0087	0.8864±0.0083	0.8589±0.0106
Self-Supervised (FT)	0.9153±0.0088	0.8979±0.0092	0.9005±0.0094	0.8744±0.0112

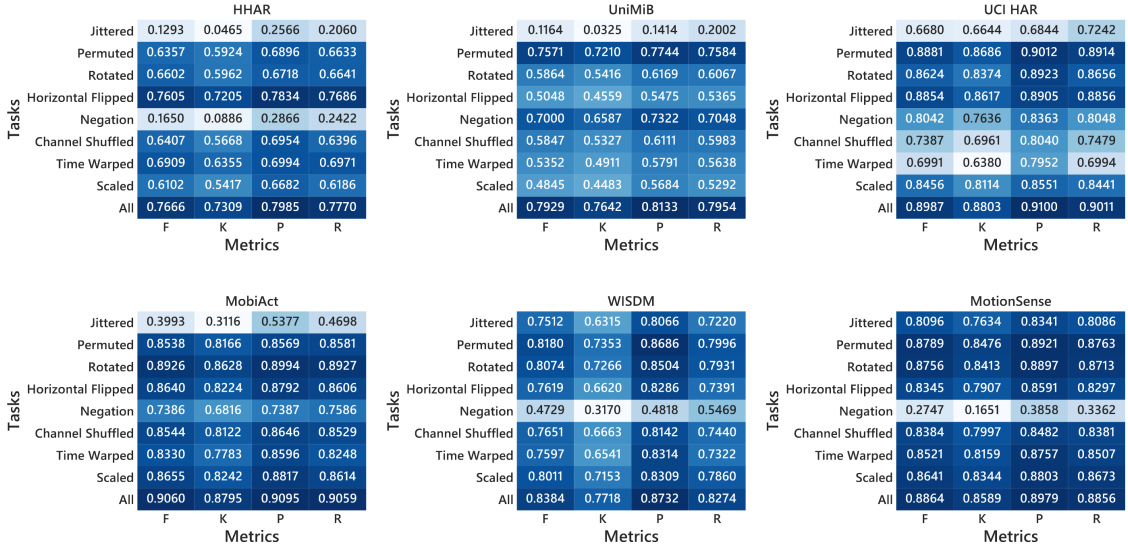


Fig. 5. Comparison of individual self-supervised tasks with the multi-task setting. The TPN is pre-trained for solving a particular task and the activity classifier is trained on-top of the learned features. We report the averaged results of evaluation metrics for 10 independent runs, where F, K, P, and R refer to F-score, Kappa, Precision and Recall, respectively. We observe that multi-task learning improves performance in all the cases with tasks such as *Channel Shuffled*, *Permuted*, and *Rotated* consistently performed better compared to other tasks across datasets.

4.3.4 Effectiveness under Semi-Supervised Setting. Our proposed self-supervised feature learning method attains very high performance on different activity recognition datasets. This brings up the question, *whether the self-supervised representations can boost performance in the semi-supervised learning setting as well or not.* In particular, can we use this to perform activity detection with very little labeled data? Intrigued by this, we also evaluate the effectiveness of our approach to semi-supervised learning. Specifically, we initially train a TPN on an entire training set for transformation prediction. Subsequently, we learn a classifier on top of the last layer's feature maps with only a subset of the available accelerometer samples and their corresponding activity labels. For training an activity classifier, we use for each category (class) 2, 5, 10, 20, 50, and 100 examples. Note that, 2-10 samples per class represent a real-world scenario of acquiring a (small) labeled dataset from human users with minimal interruption to their daily routines, hence, making self-supervision from unlabeled data of great value. Likewise, we believe, our analysis of learning with very few labeled instances across datasets is the first attempt in quantifying the amount of labeled data required to learn an activity recognizer of decent quality. For self-supervised models, as earlier, we either kept the weights frozen or only fine-tune the last *ConvC* layer.

In Figure 6, we plot the average kappa of 10-independent runs as a function of the number of available training examples. For each run, we randomly sample desired training instances and train a model from scratch. Note that, we utilize the same instances for evaluating both supervised baseline and our proposed method. The fully-supervised baseline (blue curve) shows network performance when a model is trained only with the labeled data. The proposed self-supervised pre-training technique, in particular, the version with fine-tuning of the last *ConvC* layer, tremendously improved the performance. The difference in the performance between supervised and self-supervised feature learning is significant on *MotionSense*, *UCI HAR*, *MobiAct*, and *HHAR* datasets in low-data regime (i.e. with 2-10 labeled instances per class). More notably, we observe that pre-training helps

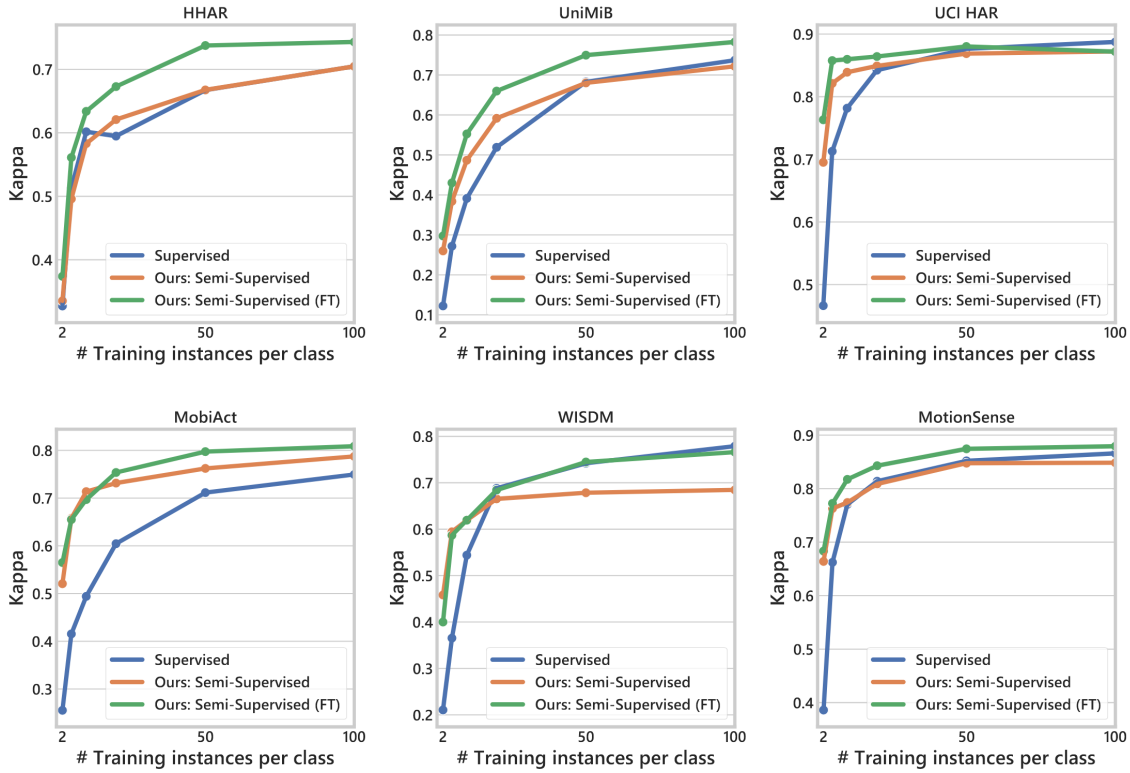


Fig. 6. Generalization of the self-supervised learned features under semi-supervised setting. The TPN is pre-trained on an entire set of unlabeled data in a self-supervised manner and the activity classifier is trained from scratch on 2, 5, 10, 20, 50, and 100 labeled instances per class. The blue curve (baseline) depicts the performance when an entire network is trained in a standard supervised way while the orange curve shows performance when we keep the transferred layers frozen. The green curve illustrates the kappa score when the last layer is fine-tuned along with the training of a classifier on the available set of labeled instances. The reported results are averaged over 10 independent runs for each of the evaluated approaches. The results with weighted f-score are provided in Figure 12 of the Appendix.

more in a semi-supervised setting when the data are collected from a wide variety of devices; simulating a real-life setting. Finally, we highlight that a simple convolutional network is used in our experiments to show the feasibility of self-supervision from unlabeled data. We believe a deeper network trained on a bigger unlabeled dataset will further improve the quality of learned representations for the semi-supervised setting.

4.3.5 Evaluating Knowledge Transferability. We have shown that representations learned by the self-supervised TPN consistently achieve the best performance as compared to other unsupervised/supervised techniques and also in a semi-supervised setting. As we have utilized the unlabeled data from the same data source for self-supervised pre-training, a next logical question that arises is *can we utilize a different (yet similar) data source for self-supervised representation extraction and gain a performance improvement on a task of interest (also in a low-data regime)?* In Table 3, we assess the performance of our unsupervised learned features across datasets and tasks by fine-tuning them on *HAHR*, *UniMiB*, *UCI HAR*, *WISDM*, and *MotionSense* datasets. For self-supervised feature learning, we utilized the unlabeled *MobiAct* dataset as it is collected from a diverse group of users that

performed twelve activities; highest among other considered datasets both in terms of the number of users and activities. This makes *MobiAct* a suitable candidate to perform transfer learning as it encompasses all the activity classes in other datasets. Of course, we do not utilize activity labels in *MobiAct* for self-supervised representation learning. We begin by pre-training a network on *MobiAct* dataset and utilize the learned weights for initialization of an activity recognition model. Moreover, the latter model is trained in a fully-supervised manner on an entire training set of a particular dataset (e.g., *UniMiB*). In comparison with supervised training of the network (from scratch), the weights learned through our technique from a different and completely unlabeled data source improved the performance in all the cases. On *WISDM* and *HHAR* our results are 3 percentage points better in terms of kappa score. Similarly, on *UniMiB* we obtained 4 percentage points improvement over supervised model, i.e. kappa score increase from 0.781 to 0.821.

Table 3. Task and Dataset Generalization: Quantifying the quality of transferred self-supervised network. We pre-train a TPN on *MobiAct* dataset with the proposed self-supervised approach. The classifier is added on the transferred model and trained in an end-to-end fashion on a particular activity recognition dataset. We chose *MobiAct* for transfer learning evaluation because of the large number of users and activity classes it covers. The reported results are averaged over 10 independent runs, where *P*, *R*, *F*, and *K* refer to Precision, Recall, F-score, and Kappa, respectively.

Dataset	Supervised (From Scratch)				Transfer (Self-Supervised)			
	P	R	F	K	P	R	F	K
HHAR	0.7624±0.0312	0.7353±0.0308	0.7276±0.0297	0.6816±0.0371	0.7816±0.0405	0.7617±0.0469	0.7549±0.0452	**0.713±0.056**
UniMiB	0.8276±0.0148	0.8096±0.0266	0.8097±0.0248	0.7815±0.0299	00.8557±0.0123	0.8444±0.0191	0.8445±0.0185	**0.8214±0.0217**
UCI HAR	0.9059±0.0133	0.8998±0.0139	0.8981±0.0148	0.8789±0.0168	0.9097±0.0129	0.9073±0.0145	0.9065±0.0152	**0.8879±0.0175**
WISDM	0.9024±0.0076	0.8657±0.0206	0.8764±0.0168	0.8211±0.0258	0.9058±0.0102	0.8907±0.0113	0.8946±0.0108	**0.8517±0.0153**
MotionSense	0.9164±0.0053	0.8993±0.0091	0.9027±0.0085	0.8763±0.011	0.9223±0.0081	0.9059±0.0132	0.9096±0.0126	**0.8843±0.016**

Further, we determine the generalization ability in a low-data regime setting, i.e., when very few labeled data are attainable from an end-task of interest. We transfer self-supervised learned representations on the *MobiAct* dataset as initialization for an activity recognizer. The network is trained in a supervised manner on the available labeled instances of a particular dataset. Figure 7 shows average kappa score of 10-independent runs of a fully-supervised (learned from scratch) and transferred models for 2, 5, 10, 20, 50, and 100 labeled instances. For each training run, the desired instances are randomly sampled, and for both techniques, the same instances are used for learning the activity classifier. In the majority of the cases, transfer learning improves the recognition performance especially when the number of labeled instances per class are very few, i.e. between 2 to 10. In particular, on *HHAR* the performance of a model trained with weights transfer is slightly lower in low-data setting but improves significantly as the number of labeled data points increases. We think it may be because of the complex characteristics of the *HHAR* dataset as it is particularly collected to show heterogeneity of devices (and sensors) having varying sampling rates and its impact on the activity recognition performance.

4.4 Determining Representational Similarity

The previous experiments establish the effectiveness of self-supervised sensor representations for activity classification that are significantly better than unsupervised and on-par with fully-supervised approaches. The critical question that arises is *whether the self-supervised representations are similar to those learned via direct supervision, i.e., with activity labels.* The interpretability of the neural networks and deciphering of the learned representations have recently gained significant attention, especially, for images (see [49] for an excellent review). Here, to better understand the similarity of the extracted representation from TPN and the supervised network, we utilize singular vector canonical correlation analysis (SVCCA) [57], saliency maps [63] and t-distributed stochastic neighbor embedding (t-SNE) [39].

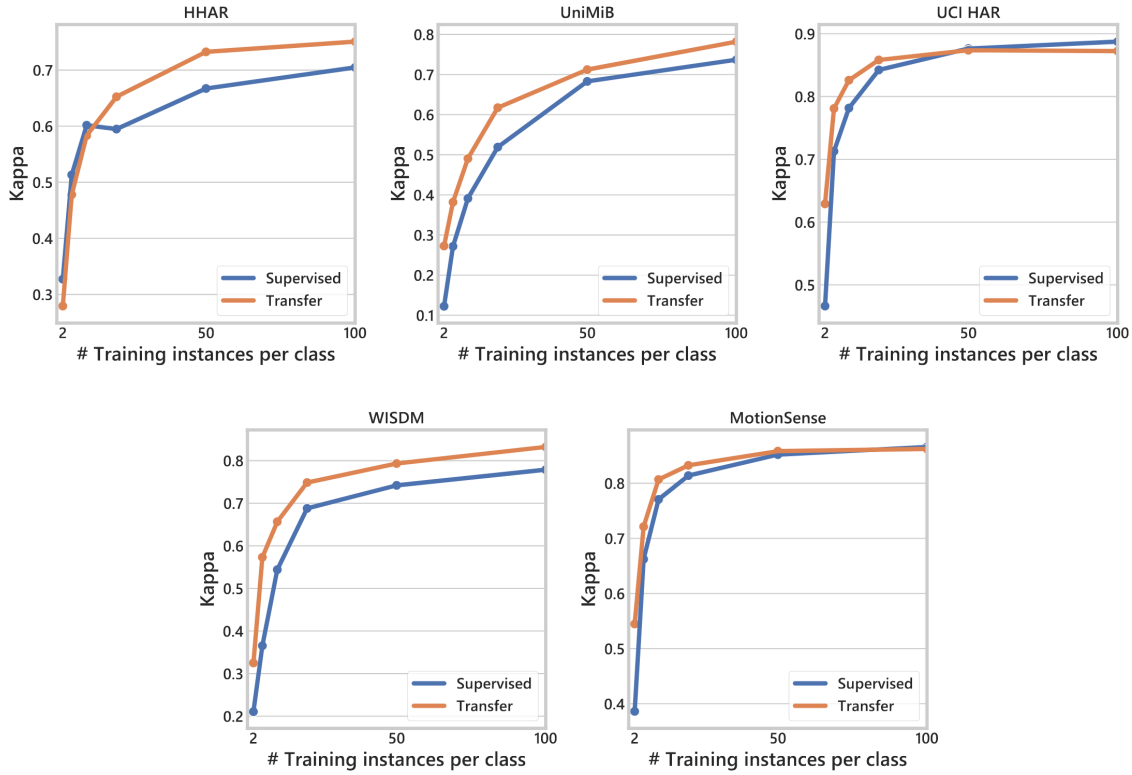


Fig. 7. Assessment of the transferred self-supervised learned features from a different but related dataset (MobiAct) under semi-supervised setting. We evaluate the performance of the self-supervised approach when different unlabeled data are accessible for representation learning but very few labeled instances are available for training a network on the task of interest. The TPN is pre-trained initially on *MobiAct* data and the activity classifier is added on-top; later an entire network is trained in an end-to-end fashion on few labeled instances. The reported results are averaged over 10 independent runs for each of the evaluated approaches when we randomly sample 2, 5, 10, 20, 50, and 100 for learning an activity classifier. The results with weighted f-score are provided in Figure **??** of the Appendix.

Insights on Representational Similarity with Canonical Correlation. The SVCCA allows for a comparison of the learned distributed representations across different networks and layers. It does so through identifying optimal linear relationships between two sets of multidimensional variates (i.e., neuron activation vectors) arising from an underlying process (i.e., a neural network being trained on a specific task) [57]. Figure 8 provides a mean similarity of top 20 SVCCA correlation coefficients for all pairs of layers for a self-supervised (trained to predict transformations) and a fully-supervised network. We averaged 20 coefficients as SVCCA implicitly assumes that all CCA vectors are equally crucial for the representations at a specific layer. However, there is plenty of evidence that high-performing deep networks do not utilize the entire dimensionality of a layer [32, 35, 45]. Due to this, averaging over all the coefficients underestimates the degree of representational similarity. To apply SVCCA, we train both the networks as explained earlier and produce activations of each layer. For a layer, where the number of neurons is larger than the layer in comparison, we randomly sample neuron activation vectors to have comparable dimensionality. In Figure 8 each grid entry represents a mean SVCCA similarity between two

layers of different networks. We observe a high correlation among temporal convolution layers trained with two different methods across all the evaluated datasets. In particular, a strong grid-like structure emerges between the last layers of the networks, which is because those layers are learned from scratch with activity labeled data and result in identical representations.

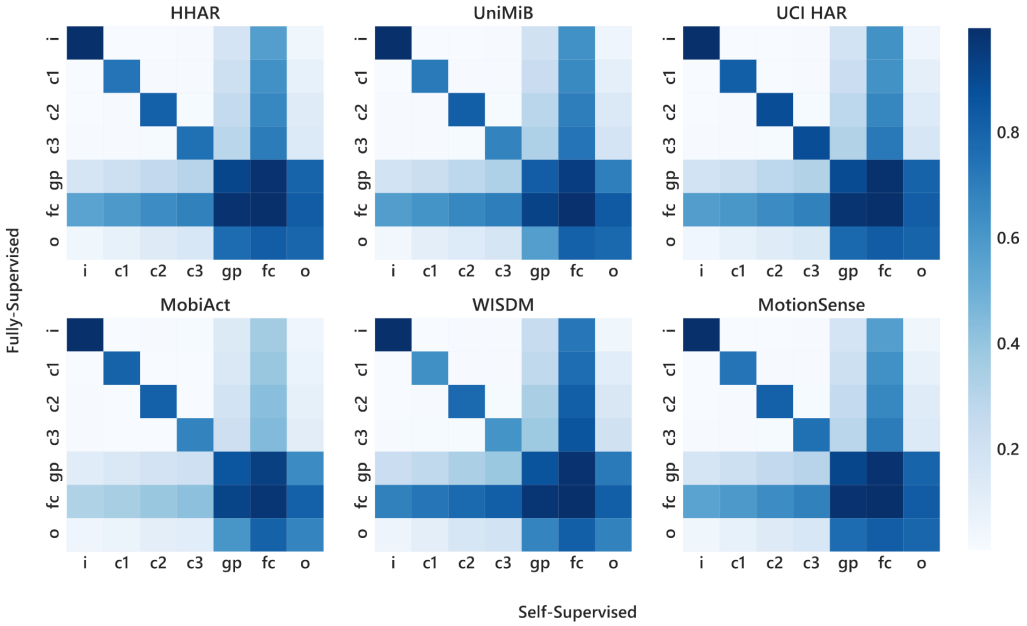


Fig. 8. CCA similarity between fully-supervised and self-supervised networks. We employ the SVCAA technique [57] to determine the representational similarity between model layers trained with our proposed approach and standard supervised setting. Each pane is a matrix of size layers × layers with each entry showing mean similarity (i.e., an average of top-20 correlation coefficients) between the two layers. Note that there is a strong relation between convolutional layers even though the self-supervised network is pre-trained with unlabeled data; showing that features learned by our approach are very similar to those learned directly via supervised learning, with activity classes. Likewise, a grid-like structure appears between the last layers of the networks depicting high similarity as those layers are always (randomly initialized and) trained with activity labels.

Visualizing Salient Regions. To further understand the predictions produced by both models, we visualize saliency maps [63] for the highest-scoring class on randomly selected instances from the *MotionSense* dataset. Saliency maps highlight which time steps largely affect the output through computing gradient of the loss function with respect to each input time step. More formally, let $x = [x_1, \ldots, x_N]$ be an accelerometer sample of length N and $C_\theta(x)$ be the class probability produced by a network $C_\theta(.)$. The saliency score of each input element x_k indicating its influence on the prediction is calculated as:

$$S_k = | \frac{\partial \mathcal{L}}{\partial x_k} |$$

where $\mathcal{L}$ is the negative log-likelihood loss of an activity classification network for an input example x.

Figure 9 provides a saliency mapping of the same input produced by the two networks for a class with the highest score. To aid interpretability of the saliency score, we calculate a magnitude of each tri-axial accelerometer sample, effectively combining all three channels. The actual input is given in the top-most pane, the magnitudes with varying color intensity are shown in the bottom panes. The dark color illustrates the regions that contribute most to the network's prediction. We observe that the saliency maps of both self-supervised and fully-supervised networks hint towards similar regions that are crucial for deciding on the class label.

Interestingly, for the *Sitting* class instance both network mainly focus on a smaller region of the input with slightly more variation in the values. We think it could be because one thing that a network learns is to find periodic variations in the signal (such as peaks and slopes). Hence, it pays more attention even to slightest fluctuation, but it decides on the *Sitting* label as the signal remains constant (before and after minor changes) which is an entirely different pattern as compared to the instances of other classes. This analysis further validates the point that our self-supervised network learns generalizable features for activity classification.

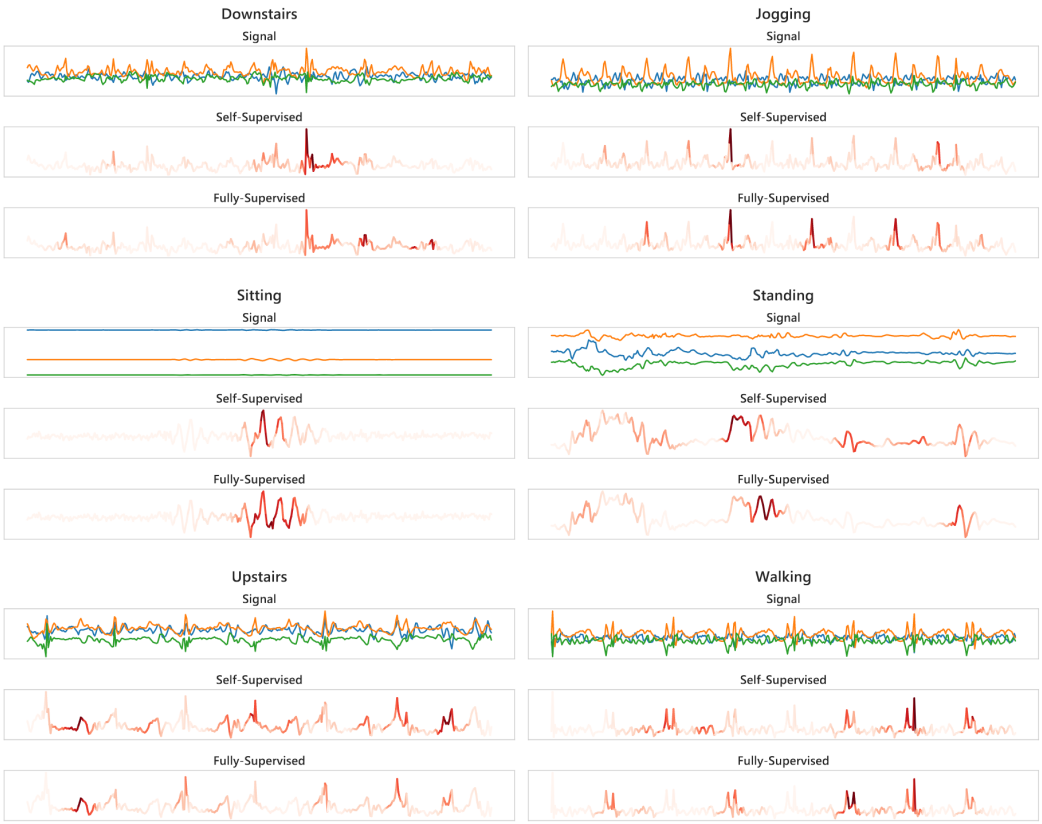


Fig. 9. Saliency maps [63] of randomly selected instances from *MotionSense* dataset. The input signal is illustrated in the top pane with the *magnitude* computed from the sample shown in the bottom panes for better interpretability. The strong colored intensities exhibit the regions that substantially affect the model predictions. The saliency mapping of both networks focus on similar input areas which shows that the self-supervised representations are useful for the end-task.

Visualization of High-Level Feature Space through t-SNE. t-SNE is a non-linear technique for exploring and visualizing multi-dimensional data [39]. It approximates a low-dimensional manifold of a high-dimensional counterpart through minimizing Kullback-Leibler divergence between them with a gradient-based optimization method. More specifically, it maps multi-dimensional data onto a lower dimensional space and discovers patterns in the input through identifying clusters based on the similarity of the data points. Here, the activations from global max-pooling layers (of both self-supervised and fully-supervised networks) with 96 hidden units are projected on to a 2D space. Figure 10 provides the t-SNE embeddings showing high semantic relevance of the learned features for various activity classes. We notice that the self-supervised features largely correspond to those learned with the labeled activity data. Importantly, the clusters of data points across two feature learning strategies are similar, e.g. in *UCI HAR*, the activity classes like *Upstairs*, *Downstairs* and *Walking* are grouped. Likewise, in *HHAR*, the data points for *Walking*, *Upstairs*, and *Downstairs* are close-by as opposed to others in the embeddings of both networks. Finally, it is important to note that t-SNE is an unsupervised technique which does not use class labels; the activity labels are just used for final visualization.

5 RELATED WORK

Deep learning methods have been successfully used in several applications of ubiquitous computing, pervasive intelligence, health, and well-being [17, 21, 38, 56, 61, 73] and eliminate the need of hand-crafted feature engineering. Convolutional and recurrent neural networks have shown dominant performance in solving numerous high-level recognition tasks from temporal data such as activity detection and stress recognition [20, 61, 68]. In particular, CNNs are becoming increasingly popular in sequence (or time-series) modeling due to their ability of weight sharing, translation invariance, scale separation and localization of filters in space and time [5, 31]. In fact, (1D) temporal CNNs are now widely used in the area of HAR (see [68] for a detailed review), but the prior works are mostly concerned with supervised learning approaches. The training of deep networks requires a huge (carefully) curated dataset of labeled instances, which in several domains is infeasible due to required manual labeling effort or can only be possible on a small-scale in a controlled lab environment. This inherent limitation of the fully-supervised learning paradigm emphasizes the importance of unsupervised learning to leverage a large amount of unlabeled data for representation learning [8] that can be easily acquired in a real-world setting.

Unsupervised learning has been well-studied in the literature over the past years. Before the era of end-to-end learning, manual feature design strategies [16] such as those that employ statistical measures have been used with clustering algorithms to discover a latent group of activities [70]. Although deep learning techniques have almost entirely replaced hand-crafted feature extraction with directly learning rich features from data, representation learning still stands as a fundamental problem in machine learning (see [8] for an in-depth review). The classical approaches for unsupervised learning include autoencoders [6], restricted Boltzmann machines [46], and convolutional deep belief networks [33]. Another emerging line of research for unsupervised feature learning (also studied in this work), which has shown promising results and does not require manual annotations, is self-supervised learning [1, 13, 58]. These methods exploit the inherent structure of the data to acquire a supervisory signal for solving a pretext task with reliable and widely used supervised learning schemes.

Self-supervision has been actively studied recently in the vision domain, and several surrogate tasks have been proposed for learning representations from static images, videos, sound, and in robotics [3, 14, 15, 18, 19, 24, 30, 34, 42, 47, 51, 52, 54, 74]. For example, in images and videos, spatial and temporal contexts, respectively, provide forms of rich supervision to learn features. Similarly, colorization of gray-scale images [30, 74], rotation classification [18], odd sequence detection [15], frame order prediction [42], learning the arrow of time [71], audio-visual correspondence [3, 52] and synchronization [27, 51] are some of the recently explored directions of self-supervised techniques. Furthermore, multiple such tasks are utilized together in a multi-task learning setting for solving diverse visual recognition problems [14]. These self-supervised learning paradigms have

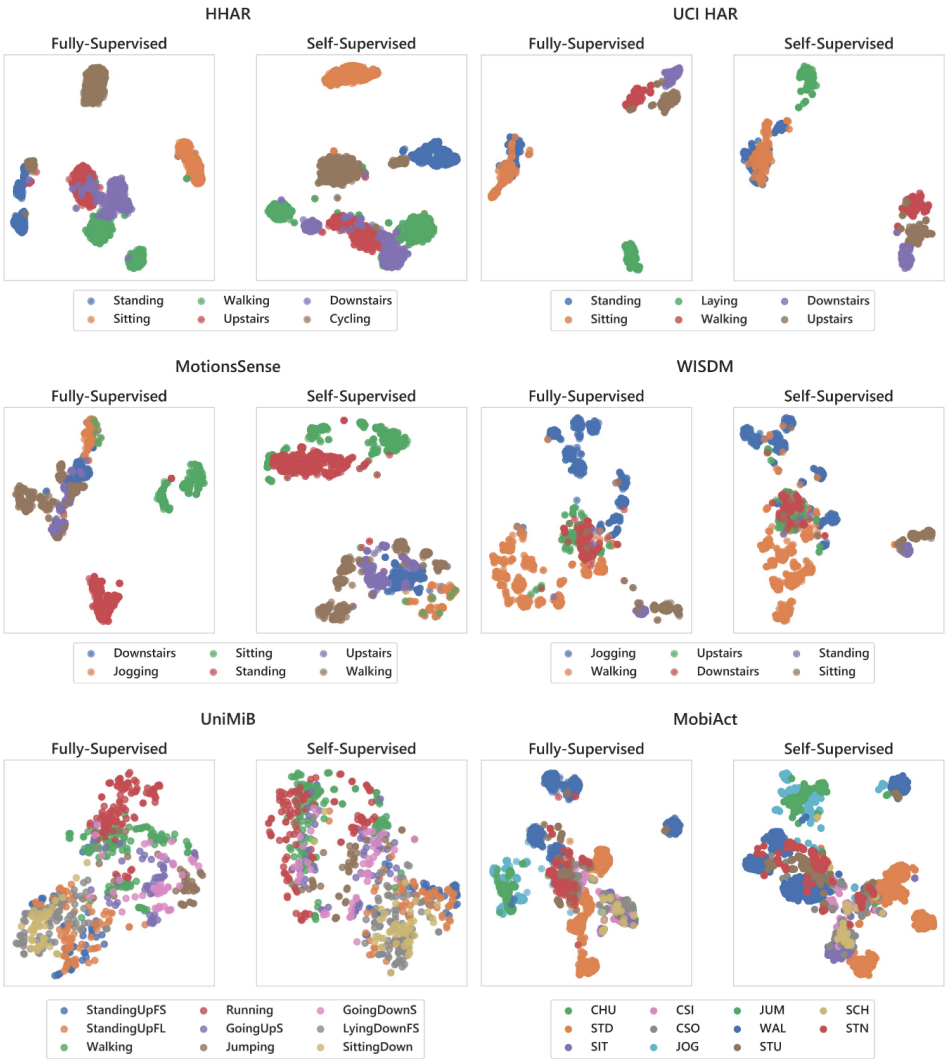


Fig. 10. t-SNE visualization of the learned representations. We visualize the features from *Global Max Pooling* layers of fully-supervised and self-supervised networks by projecting them on 2D space. The clusters show high correspondence among the representations across datasets. For instance, in *UniMiB* embeddings the samples belonging to the same class are close-by as opposed to those from a different class, such as *Running* and *Walking* are alongside each other while data point from *SittingDown* class are very far. Note that t-SNE embeddings do not use activity labels, they are only used for final visualizations.

shown to extract high-level representations that are on par with those acquired through fully-supervised pre-training techniques (e.g., with ImageNet labels) and they tremendously help with transfer and semi-supervised learning scenarios. Inspired from this research direction, we explore multi-task self-supervision for learning representations from sensory data through utilizing transformations of the signals.

Some earlier works on time-series analysis have explored transformations to exploit invariances either through architectural modifications (to automatically learn task-relevant variations) or less commonly with augmentation and synthesis. In [67] task-specific transformations (such as added noise and rotation) are applied to wearable sensor data to augment and improve the performance of Parkinson's disease monitoring systems. Saeed et al. [60] utilized an adversarial autoencoder for class-conditional (multimodal) synthetic data generation for the behavioral context in a real-life setting. Moreover, Oh et al. [48] focused on learning invariances directly from clinical time-series data with specialized neural network architecture. Razavian et al. [59] used convolution layers of varying size filters to capture different resolutions of temporal patterns. Similarly, through additional pre-processing of the original data Cui et al. [12] used transformed signals as extra channels to the model for learning multiscale features. To summarize, these works are geared towards learning supervised networks for specific tasks through exploiting invariances, but they do not address the topics of semi-supervised and unsupervised learning.

To the best of our knowledge, the work presented here is the first attempt of self-supervision for sensor representation learning, in particular for HAR. Our work differs from the aforementioned works in several ways as we learn representations with self-supervision from completely unlabeled data and without using any specialized architecture. We show that when training a CNN to predict generally known (time-series) transformations [7, 67] as a surrogate task, the model can learn features that are on a par with a fully-supervised network and far better than unsupervised pre-training with an autoencoder. We also demonstrate that the learned representations from a different (but related) unlabeled data source can be successfully transferred to improve the performance of diverse tasks even in the case of semi-supervised learning. In terms of transfer learning, our approach also differs significantly from some earlier attempts [44, 69] that were concerned with features transferability from a fully-supervised model learned from inertial measurement units data, as our approach utilizes widely available smartphones and does not require labeled data. Finally, the proposed technique is also different from previously studied unsupervised pre-training methods such as autoencoders [37], restricted Boltzmann machines [55] and sparse coding [9] as we employ an end-to-end (self) supervised learning paradigm on multiple surrogate tasks to extract features.

6 CONCLUSIONS AND FUTURE WORK

We present a novel approach for self-supervised sensor representation learning from unlabeled data with a focus on smartphone-based human activity recognition (HAR). We train a multi-task temporal convolutional network to recognize potential transformations that may have been applied to the raw input signal. Despite the simplicity of the proposed self-supervised technique (and the network architecture), we show that it enables the convolutional model to learn high-level features that are useful for the end-task of HAR. We exhaustively evaluate our approach under unsupervised learning, semi-supervised learning and transfer learning settings on several publicly available datasets. The performance we achieve is consistently superior to or comparable with fully-supervised methods, and it is significantly better than traditional unsupervised learning methods such as an autoencoder. Specifically, our self-supervised framework drastically improved the detection rate under semi-supervised learning setting, i.e., when very few labeled instances are available for learning. Likewise, the transferred features learned from a different but related unlabeled dataset (*MobiAct* in our case), further improves the performance in comparison with merely training a model from scratch. Notably, these transferred representations even boost the performance of an activity recognizer in semi-supervised learning from a dataset (or task) of interest. Finally, canonical correlation analysis, saliency mapping, and t-SNE visualizations show that the representations of the self-supervised network are very similar to those learned by a fully-supervised model that is trained in an end-to-end fashion with activity labels. We believe that, through utilizing more sophisticated layers and deep architectures, the presented approach can further reduce the gap between unsupervised and supervised feature learning.

In this work, we provided the basis for self-supervision of HAR with smartphones through a few labeled data. In the Internet of Things era, there are many exciting opportunities for future works in related areas, such as in industrial manufacturing, electrical grid, smart wearable technologies, and home automation. In particular, we believe that self-supervision is of immense value for automatically extracting generalizable representations in domains, where labeled data are challenging to acquire, but unlabeled data are available in vast quantities. We hope that the presented perspective of self-supervision inspires the development of additional approaches, specifically for the selection of appropriate auxiliary tasks (based on domain expertise) that enables the network to learn useful features to solve a particular problem. Likewise, combining self-supervision with network architecture search is another crucial area of improvement that will automate the process of optimal model discovery. Another exciting avenue for future research is evaluating self-supervised representations on an imbalanced activity dataset, where, the number of classes are high and collecting a few labeled data points for each activity class is not feasible. Finally, evaluation in a real-world setting (application deployed on real devices) is of prime importance to further understand the aspects that need improvement concerning computational, energy and, labeled data requirements.

ACKNOWLEDGMENTS

This work is funded by SCOTT (www.scott-project.eu) project. It has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 737422. This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation programme and Austria, Spain, Finland, Ireland, Sweden, Germany, Poland, Portugal, Netherlands, Belgium, Norway.
We thank Prof. Peter Baltus for a helpful discussion and anonymous reviewers for their insightful comments and suggestions. Various icons used in the figures are created by Anuar Zhumaev, Korokoro, Gregor Cresnar, Becris, Hea Poh Lin, AdbA Icons, Universal Icons, and Baboon designs from the Noun Project.

REFERENCES

[1] Pulkit Agrawal, Joao Carreira, and Jitendra Malik. 2015. Learning to see by moving. In *Proceedings of the IEEE International Conference on Computer Vision*. 37–45.
[2] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. 2013. A public domain dataset for human activity recognition using smartphones.. In *ESANN*.
[3] Relja Arandjelović and Andrew Zisserman. 2017. Objects that sound. *arXiv preprint arXiv:1712.06651* (2017).
[4] Yusuf Aytar, Carl Vondrick, and Antonio Torralba. 2016. Soundnet: Learning sound representations from unlabeled video. In *Advances in Neural Information Processing Systems*. 892–900.
[5] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271* (2018).
[6] Pierre Baldi. 2012. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning*. 37–49.
[7] Gustavo EAPA Batista, Xiaoyue Wang, and Eamonn J Keogh. 2011. A complexity-invariant distance measure for time series. In *Proceedings of the 2011 SIAM international conference on data mining*. SIAM, 699–710.
[8] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* 35, 8 (2013), 1798–1828.
[9] Sourav Bhattacharya, Petteri Nurmi, Nils Hammerla, and Thomas Plötz. 2014. Using unlabeled data in a sparse-coding framework for human activity recognition. *Pervasive and Mobile Computing* 15 (2014), 242–262.
[10] Rich Caruana. 1997. Multitask learning. *Machine learning* 28, 1 (1997), 41–75.
[11] Charikleia Chatzaki, Matthew Pediaditis, George Vavoulas, and Manolis Tsiknakis. 2016. Human daily activity and fall recognition using a smartphoneâĂŹs acceleration sensor. In *International Conference on Information and Communication Technologies for Ageing Well and e-Health*. Springer, 100–118.
[12] Zhicheng Cui, Wenlin Chen, and Yixin Chen. 2016. Multi-scale convolutional neural networks for time series classification. *arXiv preprint arXiv:1603.06995* (2016).
[13] Carl Doersch, Abhinav Gupta, and Alexei A Efros. 2015. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*. 1422–1430.

[14] Carl Doersch and Andrew Zisserman. 2017. Multi-task self-supervised visual learning. In *The IEEE International Conference on Computer Vision (ICCV)*.
[15] Basura Fernando, Hakan Bilen, Efstratios Gavves, and Stephen Gould. 2017. Self-supervised video representation learning with odd-one-out networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 5729–5738.
[16] Davide Figo, Pedro C Diniz, Diogo R Ferreira, and João M Cardoso. 2010. Preprocessing techniques for context recognition from accelerometer data. *Personal and Ubiquitous Computing* 14, 7 (2010), 645–662.
[17] Petko Georgiev, Sourav Bhattacharya, Nicholas D Lane, and Cecilia Mascolo. 2017. Low-resource multi-task audio sensing for mobile and embedded devices via shared deep neural network representations. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 3 (2017), 50.
[18] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. 2018. Unsupervised Representation Learning by Predicting Image Rotations. *arXiv preprint arXiv:1803.07728* (2018).
[19] Lluis Gomez, Yash Patel, Marçal Rusiñol, Dimosthenis Karatzas, and CV Jawahar. 2017. Self-supervised learning of visual features through embedding images into text topic spaces. *arXiv preprint arXiv:1705.08631* (2017).
[20] Nils Y Hammerla, Shane Halloran, and Thomas Ploetz. 2016. Deep, convolutional, and recurrent models for human activity recognition using wearables. *arXiv preprint arXiv:1604.08880* (2016).
[21] Awni Y. Hannun, Pranav Rajpurkar, Masoumeh Haghpanahi, Geoffrey H. Tison, Codie Bourn, Mintu P. Turakhia, and Andrew Y. Ng. 2019. Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network. *Nature Medicine* 25, 1 (2019), 65–69. https://doi.org/10.1038/s41591-018-0268-3
[22] Kazuma Hashimoto, Yoshimasa Tsuruoka, Richard Socher, et al. 2017. A Joint Many-Task Model: Growing a Neural Network for Multiple NLP Tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 1923–1933.
[23] Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 328–339.
[24] Simon Jenni and Paolo Favaro. 2018. Self-Supervised Feature Learning by Learning to Spot Artifacts. *arXiv preprint arXiv:1806.05024* (2018).
[25] Alex Kendall, Yarin Gal, and Roberto Cipolla. [n. d.]. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. ([n. d.]).
[26] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[27] Bruno Korbar, Du Tran, and Lorenzo Torresani. 2018. Cooperative Learning of Audio and Video Models from Self-Supervised Synchronization. In *Advances in Neural Information Processing Systems*. 7774–7785.
[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.
[29] Jennifer R Kwapisz, Gary M Weiss, and Samuel A Moore. 2011. Activity recognition using cell phone accelerometers. *ACM SigKDD Explorations Newsletter* 12, 2 (2011), 74–82.
[30] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. 2017. Colorization as a proxy task for visual understanding. In *CVPR*, Vol. 2. 7.
[31] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521 (27 May 2015), 436 EP –. https://doi.org/10.1038/nature14539
[32] Yann LeCun, John S Denker, and Sara A Solla. 1990. Optimal brain damage. In *Advances in neural information processing systems*. 598–605.
[33] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. 2009. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th annual international conference on machine learning*. ACM, 609–616.
[34] Hsin-Ying Lee, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. 2017. Unsupervised representation learning by sorting sequences. In *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE, 667–676.
[35] Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. 2018. Measuring the intrinsic dimension of objective landscapes. *arXiv preprint arXiv:1804.08838* (2018).
[36] Chi Li, M Zeeshan Zia, Quoc-Huy Tran, Xiang Yu, Gregory D Hager, and Manmohan Chandraker. 2016. Deep supervision with shape concepts for occlusion-aware 3d object parsing. *arXiv preprint arXiv:1612.02699* (2016).
[37] Yongmou Li, Dianxi Shi, Bo Ding, and Dongbo Liu. 2014. Unsupervised feature learning for human activity recognition using smartphone sensors. In *Mining Intelligence and Knowledge Exploration*. Springer, 99–107.
[38] Chang Liu, Yu Cao, Yan Luo, Guanling Chen, Vinod Vokkarane, and Yunsheng Ma. 2016. Deepfood: Deep learning-based food image recognition for computer-aided dietary assessment. In *International Conference on Smart Homes and Health Telematics*. Springer, 37–48.
[39] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.
[40] Mohammad Malekzadeh, Richard G Clegg, Andrea Cavallaro, and Hamed Haddadi. 2018. Protecting sensory data against sensitive inferences. In *Proceedings of the 1st Workshop on Privacy by Design in Distributed Systems*. ACM, 2.

[41] 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.

[42] Ishan Misra, C Lawrence Zitnick, and Martial Hebert. 2016. Shuffle and learn: unsupervised learning using temporal order verification. In *European Conference on Computer Vision*. Springer, 527–544.

[43] Abdel-rahman Mohamed, George E Dahl, Geoffrey Hinton, et al. 2012. Acoustic modeling using deep belief networks. *IEEE Trans. Audio, Speech & Language Processing* 20, 1 (2012), 14–22.

[44] Francisco Javier Ordóñez Morales and Daniel Roggen. 2016. Deep convolutional feature transfer across mobile activity recognition domains, sensor modalities and locations. In *Proceedings of the 2016 ACM International Symposium on Wearable Computers*. ACM, 92–99.

[45] Ari S Morcos, David GT Barrett, Neil C Rabinowitz, and Matthew Botvinick. 2018. On the importance of single directions for generalization. *arXiv preprint arXiv:1803.06959* (2018).

[46] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*. 807–814.

[47] Mehdi Noroozi and Paolo Favaro. 2016. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*. Springer, 69–84.

[48] Jeeheh Oh, Jiaxuan Wang, and Jenna Wiens. 2018. Learning to Exploit Invariances in Clinical Time-Series Data using Sequence Transformer Networks. *arXiv preprint arXiv:1808.06725* (2018).

[49] Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. 2018. The Building Blocks of Interpretability. *Distill* (2018). https://doi.org/undefined https://distill.pub/2018/building-blocks.

[50] Avital Oliver, Augustus Odena, Colin Raffel, Ekin D Cubuk, and Ian J Goodfellow. 2018. Realistic Evaluation of Deep Semi-Supervised Learning Algorithms. (2018).

[51] Andrew Owens and Alexei A Efros. 2018. Audio-visual scene analysis with self-supervised multisensory features. *arXiv preprint arXiv:1804.03641* (2018).

[52] Andrew Owens, Jiajun Wu, Josh H McDermott, William T Freeman, and Antonio Torralba. 2016. Ambient sound provides supervision for visual learning. In *European Conference on Computer Vision*. Springer, 801–816.

[53] Sinno Jialin Pan, Qiang Yang, et al. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22, 10 (2010), 1345–1359.

[54] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. 2017. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning (ICML)*, Vol. 2017.

[55] Thomas Plötz, Nils Y Hammerla, and Patrick Olivier. 2011. Feature learning for activity recognition in ubiquitous computing. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, Vol. 22. 1729.

[56] Valentin Radu, Catherine Tong, Sourav Bhattacharya, Nicholas D Lane, Cecilia Mascolo, Mahesh K Marina, and Fahim Kawsar. 2018. Multimodal deep learning for activity and context recognition. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 4 (2018), 157.

[57] Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. 2017. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *Advances in Neural Information Processing Systems*. 6076–6085.

[58] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y Ng. 2007. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning*. ACM, 759–766.

[59] Narges Razavian, Jake Marcus, and David Sontag. 2016. Multi-task prediction of disease onsets from longitudinal laboratory tests. In *Machine Learning for Healthcare Conference*. 73–100.

[60] Aaqib Saeed, Tanir Ozcelebi, and Johan Lukkien. 2018. Synthesizing and reconstructing missing sensory modalities in behavioral context recognition. *Sensors* 18, 9 (2018), 2967.

[61] Aaqib Saeed and Stojan Trajanovski. 2017. Personalized Driver Stress Detection with Multi-task Neural Networks using Physiological Signals. *arXiv preprint arXiv:1711.06116* (2017).

[62] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. 2014. CNN features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 806–813.

[63] 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).

[64] 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 mitigating mobile sensing heterogeneities for activity recognition. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. ACM, 127–140.

[65] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.

[66] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. 2014. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1701–1708.

[67] Terry T Um, Franz MJ Pfister, Daniel Pichler, Satoshi Endo, Muriel Lang, Sandra Hirche, Urban Fietzek, and Dana Kulić. 2017. Data augmentation of wearable sensor data for parkinsonâĂŹs disease monitoring using convolutional neural networks. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*. ACM, 216–220.

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

[69] Jindong Wang, Vincent W Zheng, Yiqiang Chen, and Meiyu Huang. 2018. Deep Transfer Learning for Cross-domain Activity Recognition. In *Proceedings of the 3rd International Conference on Crowd Science and Engineering*. ACM, 16.

[70] S. Wawrzyniak and W. Niemiro. 2015. Clustering approach to the problem of human activity recognition using motion data. In *2015 Federated Conference on Computer Science and Information Systems (FedCSIS)*. 411–416. https://doi.org/10.15439/2015F424

[71] Donglai Wei, Joseph Lim, Andrew Zisserman, and William T Freeman. 2018. Learning and using the arrow of time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 8052–8060.

[72] Jianbo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiaoli Li, and Shonali Krishnaswamy. 2015. Deep Convolutional Neural Networks on Multichannel Time Series for Human Activity Recognition.. In *Ijcai*, Vol. 15. 3995–4001.

[73] Shuochao Yao, Yiran Zhao, Huajie Shao, Chao Zhang, Aston Zhang, Shaohan Hu, Dongxin Liu, Shengzhong Liu, Lu Su, and Tarek Abdelzaher. 2018. Sensegan: Enabling deep learning for internet of things with a semi-supervised framework. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 3 (2018), 144.

[74] Richard Zhang, Phillip Isola, and Alexei A Efros. 2017. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *CVPR*, Vol. 1. 5.

Received February 2019; accepted April 2019

APPENDIX

Table 4. Evaluating self-supervised representation with (user-split based) 5-folds cross-validation for activity recognition. We perform this assessment based on user-split of the data with no overlap between training and test sets i.e. distinct users' data are used for training and testing of the models. The reported results are averaged over 5-folds.

(a) HHAR

	P	R	F	K
Random Init.	0.3429±0.1395	0.302±0.0465	0.2023±0.0333	0.1611±0.0536
Supervised	0.8403±0.0349	0.816±0.0518	0.8076±0.0612	0.7788±0.0624
Autoencoder	0.6068±0.2149	0.594±0.1858	0.5474±0.2182	0.5159±0.2208
Self-Supervised	0.8454±0.0378	0.8239±0.0462	0.8153±0.053	**0.7881±0.0556**
Self-Supervised (FT)	0.8439±0.0753	0.8101±0.1004	0.8038±0.1072	0.7719±0.1204

(b) UniMiB

	P	R	F	K
Random Init.	0.416±0.0307	0.3615±0.0503	0.2875±0.0722	0.2281±0.0622
Supervised	0.805±0.0095	0.7899±0.0153	0.7866±0.0165	0.7576±0.0181
Autoencoder	0.5989±0.0313	0.5743±0.0192	0.5494±0.0227	0.5009±0.0242
Self-Supervised	0.77±0.0211	0.7618±0.0191	0.7577±0.0208	0.724±0.0218
Self-Supervised (FT)	0.8396±0.0226	0.8311±0.0269	0.8285±0.0283	**0.8046±0.0309**

(c) UCI HAR

	P	R	F	K
Random Init.	0.6147±0.1845	0.5019±0.0999	0.4297±0.1141	0.3872±0.1277
Supervised	0.9068±0.0332	0.9035±0.0366	0.903±0.0377	0.8827±0.0446
Autoencoder	0.8745±0.0367	0.8485±0.0604	0.8461±0.0642	0.8161±0.0734
Self-Supervised	0.9054±0.0273	0.8919±0.0388	0.889±0.0444	0.8688±0.0472
Self-Supervised (FT)	0.9125±0.0403	0.906±0.0473	0.9046±0.049	**0.8859±0.0573**

(d) MobiAct

	P	R	F	K
Random Init.	0.4814±0.1405	0.3828±0.0417	0.3018±0.0422	0.191±0.0435
Supervised	0.9121±0.0118	0.9029±0.016	0.9043±0.0153	0.876±0.0198
Autoencoder	0.7488±0.0402	0.749±0.0398	0.7323±0.0408	0.6692±0.0542
Self-Supervised	0.8977±0.0128	0.8838±0.0133	0.8868±0.013	0.8521±0.0165
Self-Supervised (FT)	0.9185±0.0056	0.9129±0.0077	0.9127±0.0067	**0.8883±0.0101**

(e) WISDM

	P	R	F	K
Random Init.	0.6074±0.0555	0.3231±0.0958	0.3318±0.0885	0.1828±0.0623
Supervised	0.8983±0.0185	0.8799±0.0331	0.8846±0.0297	0.8359±0.0452
Autoencoder	0.6878±0.093	0.6868±0.0715	0.6671±0.0797	0.5571±0.1125
Self-Supervised	0.8711±0.0389	0.8369±0.0636	0.8462±0.0556	0.7802±0.0826
Self-Supervised (FT)	0.8842±0.0285	0.8518±0.048	0.8597±0.0409	0.7995±0.0645

(f) MotionSense

	P	R	F	K
Random Init.	0.5152±0.0997	0.4237±0.1006	0.3451±0.1069	0.2713±0.1246
Supervised	0.9332±0.0144	0.9219±0.0186	0.9242±0.0172	0.9034±0.0231
Autoencoder	0.8297±0.0658	0.8194±0.0734	0.818±0.075	0.7767±0.0892
Self-Supervised	0.9261±0.0189	0.9176±0.0195	0.9189±0.019	0.8979±0.0244
Self-Supervised (FT)	0.9476±0.0174	0.9373±0.028	0.9391±0.0254	**0.9225±0.0345**

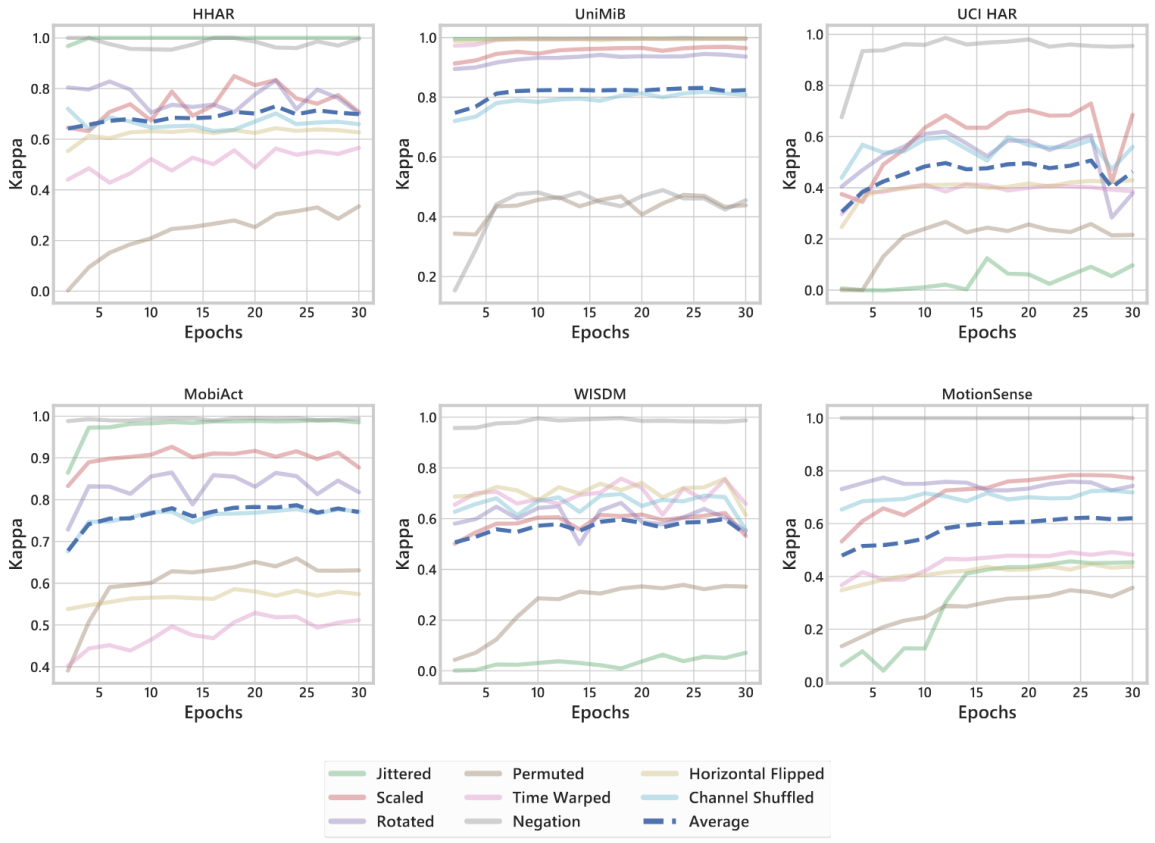
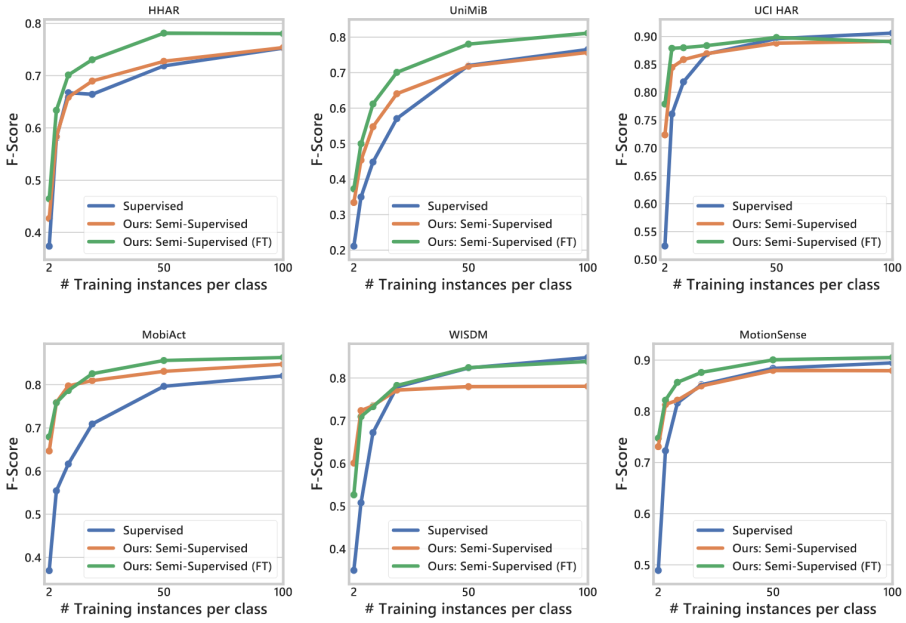
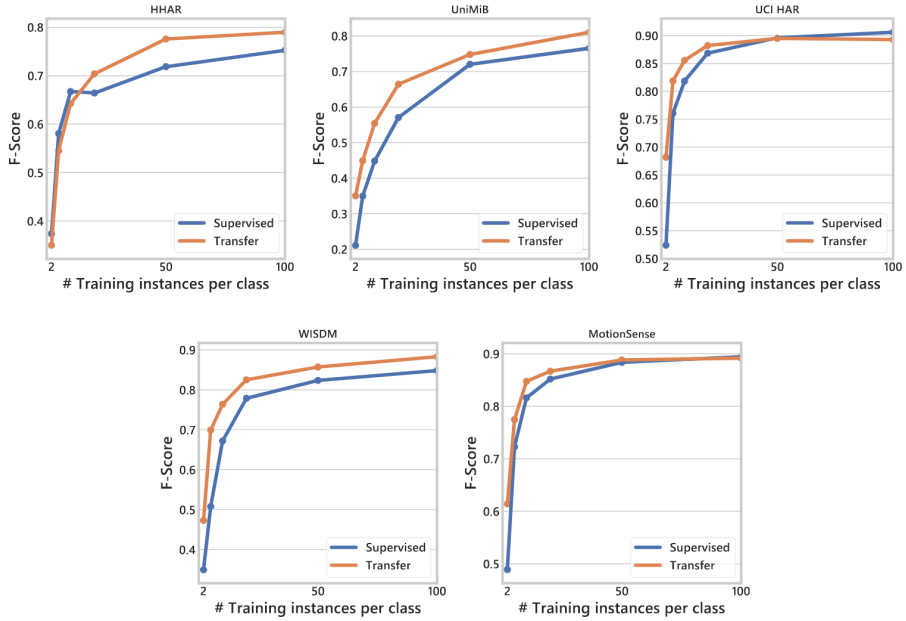


Fig. 11. Convergence analysis of transformation recognition tasks. We plot the kappa score of self-supervised tasks (i.e. transformation prediction) as a function of the training epochs. In order to produce the kappa curves, the TPN model's snapshot is saved every second epoch until the defined number of training epochs. For each saved network, we evaluated its performance on the self-supervised data obtained through processing the corresponding test sets. Note that the TPN never sees a test set data in any way during its learning phase.



(a) Weighted F-score for semi-supervised experiments, see Figure 6



(b) Weighted F-score for transferred features in semi-supervised experiments, see Figure 7

Fig. 12. The reported results are averaged over 10 independent runs for each of the evaluated approaches, for more details see Sections 4.3.4 and 4.3.5.