Pre-Training a Deep Learning Model for Human Activity Recognition Using the Source Device of Sensor Data As a Label

Benjamin Tang

Supervisor: Juan Ye

Jan 2024

1 Abstract

Building a machine learning model to perform Human Activity Recognition (HAR) historically required large labeled datasets to achieve good results. However, due to the difficult nature of collecting sensor data, the size and diversity of publicly available labeled data sets are relatively small. Recent research in other fields of machine learning has shown that pre-training techniques are able to utilize large amounts of unlabeled data to achieve similar results than their fully-supervised counterparts.

This project proposes a new pre-training technique, where the pre-trained model predicts which recording device the data originates from. By using this as a pseudo-label, the project aims to create a pre-trained model that can be fine-tuned to complete HAR tasks. This proposed technique is useful as publicly available datasets all give information about the data recording device.

The project evaluates the effectiveness of the pre-trained model on different datasets by fine-tuning the model with various amounts of downstream data and comparing the model's F1-score to a fully-supervised model and other state of the art techniques. The evaluation finds that while the proposed technique performs well on certain datasets, it fails to consistently perform better other established training techniques.

2 Acknowledgements

Thank you Erica and Fahrurrozi for being so kind and generous with your time and advice. I would not have been able to complete this without your help.

3 Declaration

I declare that the material submitted for assessment is my own work except where credit is explicitly given to others by citation or acknowledgment. This work was performed during the current academic year except where otherwise stated.

The main text of this project report is 10,126 words long, including project specification and plan.

In submitting this project report to the University of St Andrews, I give permission for it to be made available for use in accordance with the regulations of the University Library. I also give permission for the title and abstract to be published and for copies of the report to be made and supplied at cost to any bona fide library or research worker, and to be made available on the World Wide Web. I retain the copyright in this work.

Contents

1 Abstract **2**

2 Acknowledgements **2**

3 Declaration **2**

4 Introduction **5**

5 Context Survey **7**

5.1 Deep Learning Human Activity Recognition 7

5.1.1 CNN 9

5.1.2 Bi-LSTM 9

5.1.3 Transformers 10

5.1.4 Combining Architectures 10

5.2 Pre-Training 10

5.2.1 Self Supervised Learning 11

5.2.2 Transfer Learning 12

5.2.3 Semi-Supervised Learning 15

5.2.4 SelfHAR 15

6 Requirements **16**

7 Ethics **16**

8 Design and Implementation **16**

8.1 Pre-training Design 16

8.2 Datasets 18

8.2.1 HHAR 18

8.2.2 Motionsense 18

8.2.3 PAMAP2 Physical Activity Monitoring 19

8.2.4 WISDM 19

8.2.5 Daily and Sports Activities 19

8.2.6 HARTH 20

8.3 Pre-Processing 20

8.4 Architectures 21

8.4.1 Classification Head 21

8.4.2 CNN 21

8.4.3 LSTM 22

8.4.4 Transformer 22

8.4.5 CNN-LSTM 22
8.4.6 LSTM-CNN 22
8.5 Architecture Decision 23
8.6 Downstream Training 23

9 Evaluation **23**
9.1 Evaluation against baseline models 24
9.1.1 Confusion Matrices 25
9.1.2 Single Source Implementation 25
9.1.3 N-1 implementation 27
9.2 Single Source versus N-1 28
9.3 Impact of Using Variable Amounts of Data to Fine Tune Model . . . 29

10 Conclusions **31**
10.1 Further Investigation 32

A Appendix **32**
A.1 Testing 32
A.1.1 Choosing Between Architecture 32
A.1.2 Raw data for fully supervised implementation 32
A.1.3 Raw Data 32
A.2 User Manual 36
A.3 Other Appendices 38
A.3.1 Additional Confusion Matrix 38

4 Introduction

With the increased availability of sensors and the advancements in data processing technology, Human Activity Recognition (HAR) is currently undergoing a remarkable surge in interest with increasing research and applications emerging in various fields [13]. Zebhi [38] defines *activity recognition* as the "capability to recognize current activity on the basis of information acquired from a series of observations". A system performing HAR with wearable sensor data wants to be able to classify what the subject is doing given the data from the sensors on the body [33]. HAR enables precise activity monitoring, personalized recommendations, context-aware applications, and enhanced safety measures [27]. In a typical HAR system, a machine learning model is trained using data-label pairs, where the labels represent a list of potential activities, and the input data consists of recorded time-series data. By analyzing a window of data, the trained model should be able to accurately predict the specific activity that the subject is performing [33].

There are two primary approaches for HAR which are determined by the type of data being processed: vision-based and sensor-based [27] [19]. Vision-based HAR involves using computer vision techniques to analyze and interpret visual input, such as images or videos, to recognize and classify human activities [19]. For example, in a dancing video game, the systems camera can detect whether the user is making the correct dance moves. This paper, however, focuses on wearable-sensor-based HAR, where data is extracted specifically from sensors worn on a subject's body to derive the subject's activity [13]. Wearable-sensor-based HAR is particularly useful compared to vision-based HAR as it is not bound within the parameters of a vision based sensor, allowing for the subject to be monitored with fewer limiting constraints [13]. For example, in the context of fall detection, having the subject monitored by a camera means that the device is unable to report a fall if the subject isn't within the boundaries of the sensor [27]. Wearable sensors, such as accelerometers and gyroscopes, are able to collect data on body movements and orientation changes and are now made readily available on smart watches and cell phones [29]. This explosion of accessible sensors mean that there is an abundance of unlabeled time-series sensor data that can be used for HAR.

Sensor data refers to information or measurements collected by sensors that capture physical properties or phenomena in the environment [11]. In the context of wearable sensor data, it includes numerical values that represent measured quantities such as rate of change in velocity and rotation. Unlike computer vision, where data is typically in the form of images or videos that can be easily interpreted by humans [19], wearable sensor data is a continuous, multidimensional stream of information [29]. As a result, large amounts of signal processing and feature extraction are required before sensor data can be understood by humans. This abstract nature of sensor data makes post-hoc labeling almost useless. In order to produce quality labeled data, researchers and experts often need to manually label the data in real time, which is financially expensive and time consuming to do [26]. To collect and label data, small experiments are run with low levels of participants that follow fixed experimental protocol [29]. These experiments often only include a small range of activities and are not generic in nature. For example, experiments often only ask for the subjects to perform activities for a short amount of time [1], leading to data that does not properly represent real-world situations, as the data collected in the short experiments may not reflect how the activity would have been performed after it has been performed for a longer period of time [33]. Due to the inherent difficulty in collecting data, the size and diversity of labeled data sets are low [29]. This poses challenges to building a reliable HAR model, as a model trained with poor data may face challenges such as over fitting, bias and low accuracy. A model trained with a small data set may be successful in recognizing activities within it's training parameters, but struggle when faced with unseen test data [33]. A survey found that while a model trained on data collected within a controlled environment to perform well when tested in the same controlled environment, the model's accuracy dropped nearly 33% when testing in a real world environment [13]. A reliable HAR model should be able to consistently produce accurate predictions while being able to generalize to unseen data [8].

In scenarios where small amounts of labeled data are available, researchers have been finding pre-techniques to utilize unlabeled data to extract features and insights [8]. Unlabeled data refers to data that doesn't contain labels that indicate ground truths. Pre-training include transfer learning, self-supervised learning and semi-supervised learning. Transfer learning allows models to leverage knowledge from a source task or dataset, while self-supervised learning involves training models on pretext tasks using unlabeled data [41]. In semi-supervised learning, a small portion of the data is labeled, while the majority of the data remains unlabeled [29]. All approaches aim to extract meaningful representations or features from unlabeled data during pre-training, which can then be used to enhance the performance of models on downstream tasks [41][29].

The aim of this project is to utilize unlabeled data to develop a new HAR model that can achieve similar levels of success in performing HAR compared to other state of the art techniques. This project proposes an unique, pre-training learning framework that leverages a pre-text task of identifying the source of the data in order to create a pre-trained model that can be fine tuned to perform HAR. By fine-tuning a model pre-trained with a different classification task, the goal is to leverage the knowledge learned to improve the performance of HAR. This approach is powerful as it does not require additional data pre-processing during the pre-training phase, and that labels for pre-training are relatively easy to obtain. This means that large amounts of unlabeled data can be easily utilized to develop a pre-trained model that improves the model's performance on HAR, addressing the lack of data challenge HAR model faces. This pre-trained model is first trained like a normal supervised-learning task using the set of pseudo-labels, then fine-tuned using the HAR labels to allow the model to perform HAR tasks. The contributions of the project are as follows:

- Using the source of the sensor data as a pseudo-label, pre-process the data sets to use them for pre-training.
- Create pre-trained models using different state of the art machine learning techniques such as convolutional neural network (CNN), long short term memory (LSTM), transformers and combinations of these techniques.
- Pre-train and fine-tune models using the same dataset for the pre-text task and the downstream task.
- Pre-train a model with all but one of the datasets and fine-tune the model with the unseen dataset.
- Explore the relationship between the amount training data and the performance of a model by limiting the amount of data used to fine-tune the model.
- Evaluate the trained models against other existing supervised and semi-supervised training models.

5 Context Survey

5.1 Deep Learning Human Activity Recognition

HAR systems have been in development since the late 1990's, and various state of the art techniques have since been established [13]. The HAR task itself has many challenges, most notably it's intraclass similarity and variability [37]. For example, a HAR model may need to categorize between whether a subject is lying on their back or on their side, or whether a subject is running. Beyond this, HAR is limited by its lack of common definition, which can lead to unclear problem statements [4]. For example, the label "running" may vary in definition from one experiment to another, creating contradicting features when combining data. In theory, these challenges of HAR make it difficult to develop an accurate model with shallow feature learning architectures such as decision trees and support vector machines [23]. While these statistical machine learning methods have shown some success at HAR, recent experiments have shown that deep learning approaches such as neural networks are showing more effectiveness.

Deep learning is a machine learning method that is inspired by the structure of a human brain, specifically by the interconnected network of neurons [2]. Artificial Neural Networks (ANNs) are the building blocks of deep learning models, which mimic the behavior of biological neurons to process and transmit information through layers of interconnected nodes. This structure enables the model to learn complex patterns and make predictions [32]. Through the forward and back propagation process, the deep learning model is able to extract abstract features from large amounts of data to tackle complicated problems. ANN layers include an input layer, one or more hidden layers, and an output later[2]. The hidden layers within the ANN are able to learn representation, so more complex representations and features can be extracted with more layers. The forward propagation process begins with data flowing through the input layer, where each node represents a feature of the data. With each node, a weighted sum is calculated by multiplying the input data with the corresponding weights and combined with the models biases. Weights and biases are parameters with real numbers that can be tuned to influence the model's prediction and performance [14]. Each weighted sum is then passed to nodes in the first hidden layer, and similar calculations are made before proceeding to subsequent hidden layers [2]. This process continues until the transformed data reaches the output layer, which has nodes corresponding to the number of labels. This layer takes the weighted sums from the last hidden model and predicts the most probable label of the inputted data. Backward propagation first assesses the accuracy of the model by calculating the discrepancy between the predicted labels and the actual labels. This discrepancy, called loss, is then used to update the parameters of the neural network [14]. Back propagation also takes into account how changes in parameters affect the loss of the model, and appropriately adjusts them [2]. Through running forward and backward propagation multiple times, the model is then able to make predictions by learning how to effectively extract and transform information [19]. Consequently, deep learning eliminates the necessity for time-consuming and expensive handcrafted features required by statistical machine learning methods, as it can automatically extract features through its hidden layers.

These techniques have been proven to be successful in various applications, especially in HAR, as they are able to perform well with both unstructured and interconnected data [19]. As there are a large range of varying deep learning techniques, this paper consulted previous results on HAR to narrow down the architecture that this project will experiment with. Since the pre-trained model of the system will need to be fine-tuned to perform HAR tasks, it is important to find a deep learning architecture that will perform well on the downstream task [8]. According to multiple surveys, convolution neural networks (CNNs), Long Short Term Memory (LSTM) and more recently, transformers, are commonly used for HAR tasks [19] [13].

The subsections below will briefly introduce the chosen architectures and the reason why the paper explores it, while section 8.4 will detail the implementation of the chosen architectures.

5.1.1 CNN

While CNNs are traditionally known for their ability to recognize 2D-images, researchers have been utilizing 1D-CNNs for one dimensional input data [37]. CNNs have shown great success in HAR as they are able to identify and capture various salient patterns of sensor data. The convolutional filters in CNNs act as feature detectors, enabling the network to identify important temporal patterns and dependencies within the sensor data [9]. CNN's produces activation maps by using learnable kernels to convolve across the input data with each neuron in the convolutional layer only being connected to a small region of the input [2]. These activation maps represent the presence of specific features at different spatial positions [21]. The pooling operations then help to down sample the extracted features, reducing the dimensionality and enhancing the network's ability to generalize. Recent experiments have shown that the CNN models are more reliable when compared to both statistical ML methods and other deep learning methods [37].

5.1.2 Bi-LSTM

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN)[40]. Unlike traditional feed forward neural networks, RNNs have connections between nodes that form directed cycles that allow them to retain information from previous steps and use it as context for processing subsequent inputs. A limitation of standard RNNs is called the "vanishing gradient" problem, where network struggles to learn long-range dependencies in the data. LSTMs addresses this problem by introducing specialized memory cells that can retain information over longer time spans. Furthermore, there is a mechanism that selectively remembers and forgets past inputs, allowing for the network to learn more complex temporal relationships within the data.

More specifically, a bidirectional LSTM (BiLSTM) network has been shown to be able to achieve better F1-scores compared to other LSTM networks as it is able to utilize information in both forward and backward passes [40]. This is because knowing both previous and future sensor readings may improve the models ability predict the current activity. By incorporating two LSTM layers, one processing the sequence in the forward direction and the other in the backward direction, this arrangement enables the network to capture information from both past and future contexts. The forward layer processes the sequence from the beginning to the end, while the backward layer processes it from the end to the beginning.

5.1.3 Transformers

The transformer model is a recent development with the ability to effectively model and process sequential data by leveraging the self-attention mechanism [31]. This mechanism allows the model to weigh the importance of different positions by computing attention weights for each position by considering its relationships with all other positions in the sequence. These attention weights reflect the relevance or importance of each position in understanding the context of other positions within the sequence. Researchers have applied transformers to HAR due to their ability to identify connections of time-series features in both time-series direction [7]. Despite the lack of in depth research, the project attempted the transformer architecture due to its wide application range and it's ability to extract features from time-series data.

5.1.4 Combining Architectures

Researchers have recently explored the combination of different neural network architectures to create more accurate and robust HAR models [13]. One notable approach is the fusion of LSTM and CNN architectures, leveraging the strengths of both models to enhance the overall performance [35]. A common approach is to use the CNN as a feature extractor, where it processes the raw sensor data and learns spatial representations of local patterns. The output of the CNN is then fed into the LSTM, which captures the temporal dependencies and learns to model the sequential context [17]. By combining these models, the resulting architecture can harness the CNN's ability to extract spatial features from the sensor data, such as important patterns in specific sensor channels or combinations of channels. The LSTM component then takes advantage of these extracted features to model the temporal relationships and dependencies between the sensor readings. This fusion enables the model to capture both the fine-grained spatial details and the long-term temporal dynamics of the data, leading to an improved representation for activity recognition [35].

5.2 Pre-Training

Pre-training is a machine learning technique that involves training a model on a large amount of unlabeled data before fine-tuning it on a smaller labeled dataset for a specific task [29]. By learning from large amounts of unlabeled data, effective pre-training enables models to capture rich representations that generalize well across tasks and domains. It helps to alleviate the problem of data scarcity and reduces the need for extensive labeled datasets. The paper will describe various approaches below.

5.2.1 Self Supervised Learning

Self-supervised learning is also often used to leverage the abundance of unlabeled data and has been successfully applied in several applications, including pre-trained language models, generative adversarial networks and auto encoders [39]. Recently, self supervised learning has been utilized in the sensor based HAR community, such as Multi-task Self-supervision, Autoencoder, Contrastive Predictive Coding and more. Haresamudram's survey has found that HAR researchers are currently focused on finding successful architectures that do not rely on large quantities of labeled data [8]. In situations where labeled data is scarce, self-supervised models have been proven to be more high performing in different conditions than their supervised counterparts.

The high level overview of a self-supervised model includes five steps: data preparation, pretext task design, training, feature extraction and down stream training [8]. Self-supervised learning relies on unlabeled data to train the model and extract generic representations without the need for labeled datasets [29]. Labeled data refers to data that has explicit annotations or labels associated with each data instance. These labels indicate the specific categories or classes that the data represents. For example, in an image classification task, labeled data would consist of a collection of images along with their corresponding class labels like "dog" or "cat". In contrast, unlabeled data refers to raw data that lacks predefined or pre-assigned labels.

In self-supervised learning, instead of relying on labeled data provided by humans, the model learns from the inherent structure or patterns found within the unlabeled data itself [41]. This is done with pre-text tasks that guide the model to extract features and representations from the data.

The main feature of self-supervised learning is the generation of pseudo-labels from unlabeled data [10]. These pseudo-labels are dependent on the pretext task and designed to help the pre-trained model to extract features later used in the downstream task [41]. Wu *et al.* categorizes the pre-text tasks into 3 categories: contrastive, predictive and generative learning [34].

- Contrastive learning utilizes positive and negative pairs of samples to train a model to capture the similarities and dissimilarities between instances in the data [34]. Positive samples are created by applying random transformations, such as time shifting and adding noise, to the input [29], resulting in multiple versions of the same input. Negative samples are other samples that are neither the input nor derived from it. Through distinguishing between the positive and negative pairs, the model learns to capture meaningful features and structure from the data.

- Predictive learning utilizes data-label pairs to pre-train a model [36]. These pseudo-labels can be generated from four different techniques such as node property prediction, context-based prediction, self-training and domain knowledge-based prediction. Through predicting pseudo-labels, the pre-trained will extract features from the data before being downstream trained to complete their downstream task.

- Generative learning extracts features from the model through attempting to reconstruct missing parts from the original input [30]. Through predicting the missing parts of the data, the model will learn features within the data itself. The model compares the reconstructed sample and the true sample to calculate the loss for the prediction.

Depending on the down stream task and the available data, different pre-text tasks are used to help the pre-trained model extract the necessary features [41]. When done correctly, the features extracted from the pre-text task should be generic enough in which a wide range of downstream tasks can be applied to it [8]. For example, generic features can be particularly useful whether the HAR system is generalized for everyone, or if the system is user specific [13]. With a pre-trained model that has a broad spectrum of downstream training possibilities, fewer resources will be required to collect large amounts of task specific data. Once the self-supervised model has finished training, the features extracted from the deep learning model are then used as a starting point for the downstream training, where the model is fine-tuned to a different task [29].

5.2.2 Transfer Learning

Transfer learning is a machine learning technique that aims to create a reliable model by leveraging knowledge from a different but related task [28]. This process is particularly useful when the training and test data for the target task are limited. Transfer learning techniques have been widely explored and applied in top venues related to data mining, and machine learning [22]. As noted by a survey on transfer learning for activity recognition, large amounts of diverse labeled data is required to create a HAR system that performs well in a wide range of conditions [6]. In traditional machine learning approaches, models are trained and tested on the same dataset or distribution. However, we know that labeled activity data is difficult and expensive to obtain [29].

The basic idea of transfer learning is that, similar to a human brain, knowledge gained from solving a previous task can be transferred to a new task [22]. For example, when we learn what a cat looks like, we can quickly deduce what a cheetah can look like. Similarly, a model can utilize knowledge gained from solving one task to solve another. Tan et al. defines the source task as the initial task on which a model is trained using a source dataset, while the target task is the specific task that the final model is aimed to perform and it is trained with the target dataset [28]. The high level overview of transfer learning includes two steps: pre-training and fine-tuning. The goal in the pre-training task is for the label to extract knowledge of the data by solving the source task [22]. This pre-trained model is then fine-tuned by being trained with the target task's dataset to solve the target task where the model's parameters are updated. The goal in the fine-tuning phase is for the model to adapt to the target task while leveraging the general features extracted from solving the source task.

Pan describes three sub-settings of transfer learning in their survey on transfer learning [22]:

- Inductive Transfer Learning [22]: In this setting, the target task is different from the source task, regardless of whether the domains of the datasets are the same or not. A different domain can mean that the source and target data can be collected in separate contexts. Labeled data from the target domain are required to train a predictive model specifically for the target domain. Inductive transfer learning can be further categorized into two cases:
    - Labeled data is available in the source domain: This case is similar to multi-task learning, where the model aims to simultaneously learn both the source and target tasks. However, the focus of transfer learning is on achieving high performance in the target task by transferring knowledge from the source task.
    - No labeled data is available in the source domain: This case is similar to self-taught learning, where the label spaces between the source and target domains may differ, making direct use of side information from the source domain challenging.
- Transductive Transfer Learning [22]: In this setting, the source and target tasks are the same, but the domains differ. This is useful when the target domain lacks labeled data, while a substantial amount of labeled data is available in the source domain.

- Unsupervised Transfer Learning [22]: In this setting, no labeled data are available in either the source or target domains during training. The focus in unsupervised transfer learning is on solving unsupervised learning tasks in the target domain, such as clustering, dimensionality reduction, and density estimation. In this setting, the target task is different but related from the source task.

It is important to note that unlike the assumptions of traditional machine learning models, none of the sub-settings above require the training data to be representative of the data the model will encounter during deployment [28]. Large amounts of data that are not completely related to the target task can be used to train the model before fine-tuning it. This means that transfer learning can be used to solve the problem of having low levels of labeled data, as the target task does not need to be trained from scratch.

There is also four different cases of transfer learning that describes what knowledge gained from the source task is passed down to the model being fine-tuned to complete the target task. Tan et al. describes them as such [28]:

- Instances-based deep transfer learning involves a weight adjustment strategy where selected data points, called instances, from the source domain are used as supplements in the training set of the target domain [28]. These selected instances are assigned weight values based on the assumption that, despite differences between the domains, the target domain can benefit from the utilization of relevant instances from the source domain with appropriate weights.

- Mapping-based deep transfer learning involves mapping instances from both the source and target domains into a new data space where they become more similar and suitable for a unified deep neural network [28]. This approach assumes that despite differences between the original domains, the instances can exhibit greater similarity within the transformed data space.

- Network-based deep transfer learning involves reusing a pre-trained network from the source domain, including its structure and connection parameters, as a component of the deep neural network used in the target domain [28]. This approach is based on the assumption that neural networks mimic the human brain's processing mechanism and undergo iterative abstraction. The initial layers of the network can function as a feature extractor, producing versatile features that can be applied to various tasks.

- Adversarial-based deep transfer learning incorporates adversarial techniques inspired by generative adversarial networks (GANs) to discover transferable representations that can be used in both the source and target domains [28]. This approach operates under the assumption that an effective transfer requires representations that are discriminative for the primary learning task and indistinguishable between the source and target domains. By leveraging adversarial training, the model aims to learn representations that capture relevant information for the task while being domain-agnostic.

The different cases show that not all knowledge gained from the source task is useful in the target task. In some situations, knowledge from the source task may be detrimental to the performance of the target task. Ultimately, the goal of transfer learning is to provide the target model additional knowledge to better solve the target task.

5.2.3 Semi-Supervised Learning

Semi-supervised learning is a machine learning technique that is a combination of supervised and unsupervised learning [24]. The main characteristic of this technique is that the model is built with limited labeled data and a larger amount of unlabeled data by keeping a portion of the labels within the labeled dataset [39]. By utilizing both unlabeled and labeled data, semi-supervised learning aims to address the drawbacks of supervised and self-supervised learning [24]. However semi-supervised learning is limited by it's restricted scalability in addition to the technique is not being proven with real life data [29][24].

5.2.4 SelfHAR

SelfHAR is a training pipeline that utilizes unlabeled data, allowing deep learning modules to generalize to unseen scenarios by combining self-training and self-supervised learning techniques [29]. In the context of HAR, labeled data refers to sensor data that has been manually annotated with activity labels, while unlabeled data refers to sensor data without any corresponding labels. This paper shows that by using unlabeled data for feature extraction, the model produces robust results, especially in cases where limited data are available.

The aim of SelfHAR's methods is to tackle the lack of diversity and size of data sets addressed above. In the past, individual pre-training has been used to increase the quantity and diversity of the learning models. The goal of SelfHAR is to fully utilize the potential of pre-training by combining self-supervised and semi-supervised training techniques. SelfHAR's methodology consists of two main steps. First, a base model is trained using the available labeled data. This model is then used to predict the activity labels for the unlabeled data. The predicted labels are treated as pseudo-labels and used to augment the labeled dataset. In the second step, a new model is trained using the augmented labeled dataset, which includes the original labeled data and the pseudo-labeled data. This process is repeated iteratively, with the newly trained model being used to generate pseudo-labels for unlabeled data in each iteration.

Models trained on SelfHAR's training pipeline shows state of the art results, out performing other supervised and semi-supervised approaches. By combining pre-training techniques, the model is able to be trained on more internally and externally diverse data without increasing the complexity of the models during inference. The research highlights the potential of pre-training and combining techniques for leveraging unlabeled data.

6 Requirements

The project was developed using Keras TensorFlow, with training and testing done on a NVIDIA GeForce RTX 3060 and two Quadro P6000s.

7 Ethics

The project uses publicly available data sets which do not contain information about the users of the trials.

8 Design and Implementation

The section below will describe the technical specifications of this project.

8.1 Pre-training Design

This project's pre-training model takes inspiration from both transfer learning and self-supervised learning. Since the overall architecture more closely resembles that of self-supervised learning, the project will continue to use terms typically associated with self-supervised learning architectures.

The pre-text task of this project is to predict which recording device the data come from given a window of sensor data, and the downstream task is human activity recognition. In order for the model to predict whether a window sensor data has come from a certain device—for example deciphering between accelerometer data coming from a right thigh verses a right arm—the model needs to learn useful signal characteristics from the data [8]. When analyzing different publicly available datasets, it has been found that researchers also provide information about the device in which the data is collected. This is useful as large amounts of unlabeled data by can now be easily labeled by simply knowing which device the data has come from. There is also an added benefit that different datasets often utilize different devices to collect data, which means that the diversity of labels will be large when combining data from different datasets. While in self-supervised learning, the pseudo-labels created for the pre-text task is often self-generated from unlabeled data [41]. This pre-text task deviates slightly from a typical self-supervised learning framework, instead it closely resembles how a source task functions in inductive transfer learning, as the pseudo-labels used are not generated by the model itself. Instead, the labels are information about the data itself and are used to form a classification task. If the pre-text task is able to extract useful knowledge, the downstream model will perform better than its fully supervised counterpart and will only need small amounts of data to have accurate results [41].

There are two implementations of the pre-trained model which aim to evaluate whether the pre-text task is able to pass useful knowledge to the downstream task.

The first implementation only uses one dataset, referred to in this project as the Single Source (SS) implementation. The dataset is split into training, validation and testing sets in the pre-processing step. The testing set is used to test both the pre-text task and the downstream HAR task. This implementation is designed to evaluate whether the pre-trained model is able to extract useful features with small amounts of training data. It is important to note that the model is trained twice on the same datasets, first with the pseudo-labels and second with the HAR labels. This, in theory, should allow for the model to extract more insightful features on the specific dataset.

The second implementation uses all but one of the datasets to be split into training and validation sets, with the one remaining dataset used to test the downstream task. This project will call this the N-1 implementation. As mentioned above, the benefits of unsupervised learning is the ability to utilize large amounts of data. All of the data from each dataset and their respective pseudo-labels will be concatenated for this implementation. Since different datasets often place recording devices on similar places, this pre-text task will require the pre-training model to be able to differentiate between more similar data signals and, by proxy, differentiate between datasets. For example, data collected from an IMU device on a wrist may have similar features as data collected from a smart watch. The additional features extracted from this model may be beneficial in helping the HAR task to differentiate between similar activity labels. However, the model runs into the risk of having extracted features that are not generic. If this implementation is able to extract generic features, then the model should be able to perform HAR at a high level despite the unseen testing data. Comparing the two implementations should give us insight on the pre-text task ability to extract insightful and generic features from the pre-text dataset.

The pre-training task follows the architecture proposed in the section above using the pseudo-labels as the output in the output layer of the deep learning module.

Training this model is the same as training a fully supervised model with forward and backward propagation. Once source task is completed, the output layer is then removed and a new output layer containing the labels for the HAR task is inputted.

8.2 Datasets

Five datasets were used in this study, all of which include information about the device that are used to record the data and activity labels. There is a combination of data collected in controlled experiments and in real life situations.

8.2.1 HHAR

The Heterogeneity Dataset for Human Activity Recognition (HHAR) dataset [3] is a dataset that collects data from 9 subjects. The recording scenario is designed to be realistic, aiming to generate data that reflect real life. The users take two different routes when biking and walking, and two different staircases when climbing up and down stairs. The readings are recorded while users are performing scripted activities in no specific order, carrying both smartwatches and smartphones. The activities included in the dataset are biking, sitting, standing, walking, stair up and stair down.

The dataset includes recordings from four different smartwatches (two LG watches and two Samsung Galaxy Gears) and eight different smartphones (two Samsung Galaxy S3 mini, two Samsung Galaxy S3, two LG Nexus 4 and two Samsung Galaxy S+). The accelerometer and gyroscope found in the devices were sampled at the highest frequency allowed by each device. The dataset specifies the device in which the data is collected from, giving the pre-trained model a wide range of labels to classify in the pre-text task.

8.2.2 Motionsense

The motionsense dataset [18] is collected to analyze personal attributes from the sensor data. Data is collected from 14 male and 10 female subjects ranging from the age of 18-46. The experiments ran in this research are semi-controlled, where the participants were asked to wear flat shoes to perform 6 specified activities by following a specific route around Queen Mary University of London's Mile End campus. The 6 specified activities are: walking downstairs, walking upstairs, walking, jogging, sitting, and standing.

The experiment collects data using the Crowdsense app installed on an Iphone 6 that is placed in the subjects' trouser's front pocket. The data is collected at 50Hz.

8.2.3 PAMAP2 Physical Activity Monitoring

The PAMAP2 Physical Activity Monitoring dataset [25] is a dataset collected from 9 subjects wearing three inertial measurement units (IMUs) and a heart rate monitor performing 24 different physical activities. The subjects are predominantly male with one female subject, with their age ranging from 23-32. The dataset description says that the subjects were asked to follow a protocol that included 12 different activities, with the 6 other activities being optional. However there are 24 activity labels within the dataset. The labeled activities are lying, sitting, standing, walking, running, cycling, nordic walking, watching tv, computer work, car driving, ascending stairs, descending stairs, vaccume cleaning, ironing, folding laundry, house cleaning, playing soccer, rope jumping and transient. The data labeled as transient is discarded as noted by the researchers.

This dataset is quite problematic as the subjects do not each perform all of the activities, in addition to there being missing values within the dataset. Along with the imbalance of data per activity label, there is also an imbalance of the number of samples each user has. However, despite these flaws, this dataset is useful in this experiment, especially in the downstream testing phase, as it will show whether the pre-trained model is generic enough to operate on imperfect data.

8.2.4 WISDM

The Wireless Sensor Data Mining (WISDM) dataset was created for the paper Activity Recognition using Cell Phone Accelerometers [12]. The data is collected from 29 subjects by placing an Android phone in their front trousers pocket and recording the data on a custom application. There are 6 labels, which include walking, jogging, upstairs, downstairs, sitting and standing. The data is recorded at 20Hz.

8.2.5 Daily and Sports Activities

The Daily and Sports Activities dataset (DSAD) [1] is a dataset collected in a controlled experiment in three controlled environments. Data is collected from four male and four female subjects between the age of 20 and 30. The experiment contains 19 activities, in which the subject were asked to perform the activities for five minutes with their own interpretation. The activities are listed as: sitting, standing, lying on back and on right side, ascending and descending stairs, standing in an elevator still, moving around in an elevator, walking in a parking lot, walking on a treadmill with a speed of 4 km/h (in flat and 15 deg inclined positions), running on a treadmill with a speed of 8 km/h, exercising on a stepper, exercising on a cross trainer, cycling on an exercise bike in horizontal and vertical positions, rowing, jumping, and playing basketball. Since all of the data are collected for five minutes, the dataset is very balanced as there is an equal number of data for each activity.

The data is collected from 5 separate units each with an accelerometer, a gyroscope and a magnetometer. The units are placed on the torso, right arm, left arm, right leg and left leg. The sensors used are collected at 25Hz, in which the signals are divided into 5 second segments. The wide range of data source gives the pre-trained model more potential labels to be trained on.

8.2.6 HARTH

The Human Activity Recognition Trondheim (HARTH) dataset [16] consists of recordings from 22 participants who wore 2 3-axial Axivity AX3 accelerometers for approximately 2 hours while going about their daily activities. One sensor was attached to the front right thigh, and the other was attached to the lower back. The data is sampled at a rate of 50Hz. Video recordings from a chest-mounted camera are used to annotate the activities performed by the participants frame-by-frame. Due to the unsupervised nature of the experiment, the data has an imbalance of labels.

The dataset includes annotations for various activities, such as walking, running, shuffling, ascending and descending stairs, standing, sitting, lying down, cycling while sitting, cycling while standing, and inactive cycling while sitting and standing.

8.3 Pre-Processing

The data pre-processing stage follows a process similar to SelfHAR [29] in order to facilitate a fair comparison. During data extraction, the device name is collected as a pseudo label, in addition to the x, y, z axis and activity labels. For example, for the Daily and Sports Activities data set, data coming from the thigh accelerometer will have a value of “thigh acc” in the column “device” [1]. Data is then split into training, validation and testing sets. It is important that the testing data is not used in pre-training and separated from the training data to prevent data leaks. Z-normalization is then applied onto each of the sensor channels for the training data. Following this, data is segmented into sliding windows. Each sliding window contains 400 time stamps and 3 channels (x, y, z), with each window sharing 75% overlap with the next window. The overlap is larger than SelfHAR’s model as preliminary findings shows that it performed better than with having a smaller overlap. Depending on whether the data is pre-processed for the pre-text task or the HAR task, the label is either the device name or the activity. Like SelfHAR, the sampling frequency of the different datasets is not standardized as it did not improve the performance. Instead of directly splitting data, users were instead split into training and testing sets.

This project decided to only use the accelerometer data as it allows for a better comparison with SelfHAR.

8.4 Architectures

Before evaluating the pre-training model, the project explores different potential architectures in which the model will be built on. The different architectures are evaluated by running the HHAR dataset through the pre-training and downstream training tasks. The outputs of the downstream tasks will be analyzed and the most reliable architecture, determined by the highest weighted F1-Score, will be used in the evaluation section of the paper. Each of the architectures described will not include an output layer, instead each section will describe a core model in which a classification head will be attached.

8.4.1 Classification Head

This classification head is shared by all of the models and taken from the full HAR classification head found in SelfHAR [29]. The core model is first passed into a dense layer with 1024 units and then activated by a rectified linear unit (ReLU) layer. This is then attached to an output dense layer with the same number of units as the number of classification labels. Lastly, a softmax function is applied to the previous dense layer to activate it.

8.4.2 CNN

The CNN architecture is taken from the paper Multi-task self-supervised learning for human activity detection that also explored self-supervised learning for HAR [29][26]. The main model architecture comprises three 1D convolutional layers. The first layer has 32 filters with a kernel size of 24, the second layer has 64 filters with a kernel size of 16, and the third layer has 96 filters with a kernel size of 8. All layers have a stride of 1. The ReLU activation function is applied to the output of each convolutional layer. To mitigate over fitting, L2 regularization is applied to the weights with a regularization factor of 0.0001.

A drop-out layer is inserted between each pair of convolutional layers, with a dropout rate of 0.1. This drop-out layer helps prevent the model from relying too heavily on specific features and encourages robustness.

To consolidate the outputs of the convolutional layers, a global 1D maximum pooling layer is connected to the final convolutional layer. This pooling layer extracts the maximum value across each filter's activation map, reducing the dimensionality and capturing the most salient features [29].

8.4.3 LSTM

The bidirectional LSTM architecture consists of two LSTM cells, one for the forward pass and the other for the backward pass [40]. The outputs of these two layers are concatenated to create a single combined output. Each of the LSTM layers implemented in Keras will have 96 units, with a ReLU activation layer attached to the output of the bidirectional LSTM layer. Following the bidirectional LSTM layer, an activation layer with a sigmoid function is attached to introduce non-linearity. A dense layer with 100 units is then attached to the activation layer.

8.4.4 Transformer

The transformer model is based on an online tutorial on the Keras website [20]. The transformer encoder is created using multi-head attention and convolutional layers. The multi-head attention is an implementation based on Vaswani's paper "Attention is All You Need" [31], where the key dimension is set to 512, number of heads set to 16 and the dropout probability set to 25%. This is followed by another dropout layer before the output is normalized. The feed forward network layer is attached after the transformation layer using a stack of convolution and dropout layers. The convolution layers perform linear transformation to further process the outputs.

8.4.5 CNN-LSTM

The CNN-LSTM model is inspired by the architecture described in the paper Sravan's "A multi branch CNN-BiLSTM model for human activity recognition using wearable sensor data" [5]. The input is first passed through a dense layer with 300 units and a ReLU activation function. Two 1D convolutional layers follow, with 64 and 128 filters, respectively, using a kernel size of 3 and a ReLU activation function. Max pooling is applied with a pool size of 4 and a stride of 2. Two additional 1D convolutional layers with 32 filters, a kernel size of 5 and a ReLU activation function are used. Another ax pooling layer is applied with the same parameters. All of the convolution layers utilize L2 regularization with a regularization factor of 1e-4. Dropout layers with rates of 0.5 and 0.4 are added. A bidirectional LSTM layer with 96 units is then attached to the CNN layers. A dense layer with 100 units and softmax activation layer is attached to the end to be used for classification.

8.4.6 LSTM-CNN

This model is different from the CNN-LSTM model in that the input is passed through LSTM layers before being passed through convolutional layers. This architecture has shown success in HAR applications and is based on Xia's paper "LSTM-CNN

Architecture for Human Activity Recognition" [35]. The architecture passes the input through two LSTM networks containing 32 neurons each. The output is then passed onto an 1D convolutional layer with 64 filters, a kernel size of 5, ReLU activation, stride of 2 and L2 regularization is applied to the weights. A dropout layer with a rate of 0.1 helps prevent overfitting. Following that, max pooling is performed with a pool size of 2, valid padding, and a stride of 2. Another 1D convolutional layer with 128 filters, kernel size of 3, ReLU activation, stride of 1, and L2 regularization is applied to the weights. This is then passed through a max pooling layer before being passed onto the classification head.

8.5 Architecture Decision

The HHAR dataset is used as a sample dataset to help determine which model to fully evaluate. This is done by using the process described in section 9.1.1, where the weighted F1-score is calculated from the downstream task. Seen from figure 13. the initial experiments show the CNN architecture as the most reliable architecture. Figure 13 shows the raw data and results from the preliminary evaluation.

8.6 Downstream Training

After the pre-trained model finishes training with the pseudo-labels, downstream training utilizes the features extracted from the model in order to complete the HAR task. This is done by removing the classification head of the pre-trained model, and attaching a new classification head mapped with the activity labels. This new model is then fine-tuned by being trained with the training data set aside in the data pre-processing phase.

9 Evaluation

Since this project is largely based on the SelfHAR model, the evaluation method is similarly inspired by the paper SelfHAR [29]. This paper will compare all of its models against SelfHAR's evaluation on the HHAR, MotionSense and WISDM dataset.

This paper uses a weighted F1-score to evaluate the performance of the models. A normal F-1 score is a metric that uses the precision and recall of the model to provide a balanced assessment of the model's performance [15]. The precision of a task is the number of true positives divided by the sum of true positives and false positives; the recall of the task is the number of true negatives divided by the sum of true negatives and false negatives. True positives are the cases where the model correctly predicts the positive class, while true negatives are the instances where the model correctly predicts the negative class. False positives, on the other hand, are the instances where the model incorrectly predicts the positive class when the true class is negative, while false negatives are cases where the model incorrectly predicts the negative class. However, using a normal F-1 score may be misleading when the distribution of classes is not equal. For example, when the data is skewed towards a single label, the model that predicts all samples to be the majority class will achieve a high F-1 score. The weighted F1 score takes into account the class distribution by computing the average F1 score, weighted by the number of samples of each class. This means that the F1 score of each class contributes proportionally to its representation in the dataset, giving a fair assessment of overall model performance. This evaluation metric gives us a more comprehensive understand of the model's performance within the context of this project as some of the datasets are imbalanced.

9.1 Evaluation against baseline models

Table 1 below compares the weighted F1-score of the project's transfer learning model against the baseline set by SelfHAR [29]. The fully supervised metric is taken from SelfHAR.

Table 1: Performance comparison of different methods on various datasets

Dataset	**Fully Supervised**	**Transformation Discrimination**	**SelfHAR**	**SS**	**N-1**
HHAR	0.7282	0.7961 (+6.79%)	0.7846 (+5.64%)	0.8185 (+9.03%)	0.7741 (+4.59%)
Motion Sense	0.9275	0.9295 (+0.20%)	0.9631 (+3.56%)	0.9464 (+1.89%)	0.9467 (+1.92%)
WISDM	0.8906	0.8948 (+0.42%)	0.9081 (+1.75%)	N/A	0.8385 (-5.21%)

Table 2 below compares the weighted F1-score of the project's own supervised model versus the proposed model. The supervised model was trained using the HAR labels and the CNN architecture. WISDM is missing the SS implementation as the data was collected on a single type of device, which meant that the pre-training could not be implemented.

Table 2: Performance comparison of different datasets

Dataset	**Fully Supervised**	**SS**	**SS vs Fully Supervised**	**N-1**	**N-1 vs Fully Supervised**	**N-1 vs SS**
PAMAP	58.48%	63.13%	4.65%	58.35%	-0.13%	-4.79%
HHAR	76.76%	81.85%	5.08%	77.61%	0.84%	-4.24%
Motionsense	94.07%	94.07%	0.00%	94.79%	0.72%	0.72%
Harth	87.43%	84.33%	-3.10%	86.27%	-1.16%	1.93%
DASD	75.05%	62.78%	-12.27%	78.56%	3.51%	15.78%
WISDM	83.37%	N/A	N/A	84.61%	1.24%	N/A

9.1.1 Confusion Matrices

Below are the confusion matrices collected from each implementation of the proposed architecture.

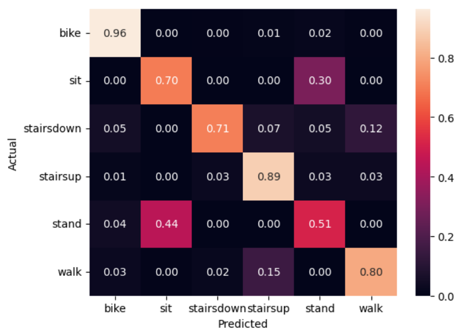


Figure 1: Confusion Matrix of HHAR (SS)

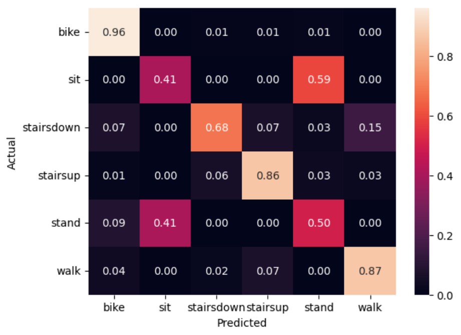


Figure 2: Confusion Matrix of HHAR (N-1)

9.1.2 Single Source Implementation

To determine whether the model is able to be successful in scenarios with small amounts of data, the model is trained on a single dataset split into a training and testing set. The training set will use 80% of the users, leaving the remaining 20% being part of the testing set. Once the model is done training on the pre-text task, the model is first evaluated with the testing data.

The model is then fine-tuned to complete HAR using only a percentage of the training data to determine whether the amount of data used to downstream train

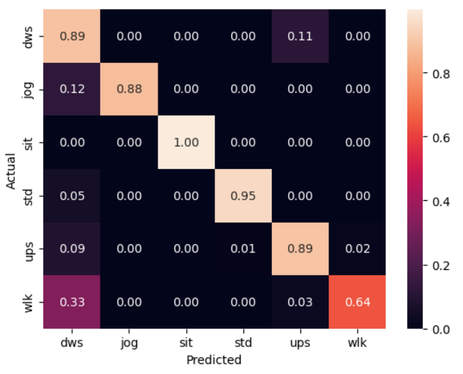


Figure 3: Confusion Matrix of Motionsense (SS)

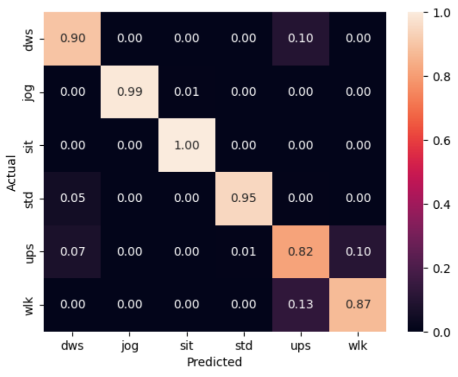


Figure 4: Confusion Matrix of Motionsense (N-1)

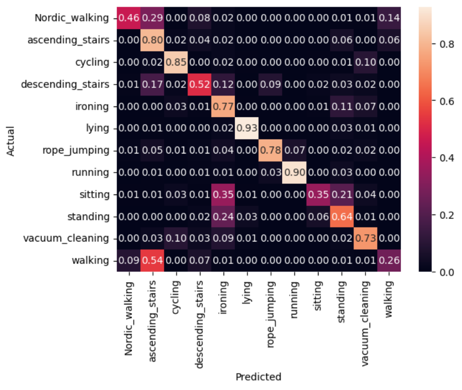


Figure 5: Confusion Matrix of PAMAP (SS)

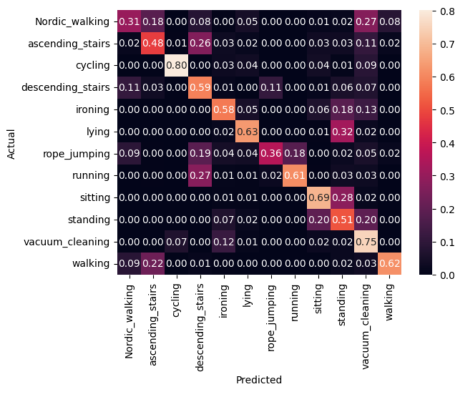


Figure 6: Confusion Matrix of PAMAP (N-1)

affects the accuracy of the HAR task. This is done by limiting the number of users employed to fine-tune the model. Section 9.2 will further discuss this.

The pre-text task of predicting the source location of the data on the body has shown mixed results compared to the fully supervised model. As seen in Table 2, the proposed model performs worse at predicting activity labels for the PAMAP and Harth datasets while only showing improvements for the HHAR and DASD datasets. When compared to SelfHAR in Table 1, the proposed model performs better at identifying activities on the HHAR dataset while falling short on the motionsense dataset.

A hypothesis on the model's superior performance on the HHAR dataset is that the pseudo-labels are more detailed. Unlike other datasets where the model is only trained on predicting where the sensor is on the body, the HHAR dataset has additional pseudo-labels on the types of device being used. This may mean that the pre-trained model is able to extract additional features, hence the improvement of performance. Furthermore, the model's poor performance on other datasets make it unclear whether the pre-texxt task is fully capable of extracting generic features. This will be discussed further in section 9.1.3. While the model's performance on the HHAR dataset is promising, the more intricate pre-training process featured in SelfHAR likely captures additional nuances and contextual information, which could lead to consistently improved performance. Despite its relative success, the model pre-trained with the pre-text task focused solely on source location may lack the ability to generate features that the SelfHAR approach is capable of extracting.

9.1.3 N-1 implementation

Since the pre-training models should work best with large amounts of training data [29], the paper proposes that if all of the datasets were concatenated together to create labels and training data for the pre-text task, the model could be trained to extract additional features. For example, to perform HAR on HHAR, all of the project's datasets—except for HHAR—will be concatenated together to train the model to solve the pre-text task. Once the model has been pre-trained, the HHAR dataset is then split into training, validation and testing sets. After attaching a new classification head, the model is fine-tuned using the HHAR training data. To evaluate the project's ability to be fine-tuned with small amounts of data, the same process as the previous section by limiting the number of training data the model will be used.

As seen in Table 2, the proposed method of using other datasets to extract features from sensor data produces mixed results. While the weighted F-1 score of Harth and DASD improves significantly compared to the SS implementation , the motionsense dataset only improves by less than 1%. Furthermore, PAMAP and HHAR's weight

F-1 score is significantly worse.

As seen in Table 1, both implementations fall short of SelfHAR's F1-score.

9.2 Single Source versus N-1

With the observed mixed results, the proposed implementations of Self Supervised learning may not be robust enough. Compared to other pre-text tasks, which often involves augmenting the data to create labels, the labels are pre-generated [41].

In the PAMAP and HHAR dataset, the model pre-trained with a larger dataset performs worse than the model that is pre-trained and fine-tuned on the same dataset. This can be seen in both the weighted F1-score and also the confusion matrices. While both implementations have similar inaccuracies in predicting similar activities, it is clear that the N-1 implementation makes those mistakes more often. On the HHAR dataset in Figure 2, the N-1 model mislabels 'sit' with 'stand' 29% more frequently than the SS model. Furthermore as seen in Figure 6, the N-1 model mislabels more activities compared to the SS model. This may mean that the pre-text task is not able to train the model to extract generic features from the concatenated data, ultimately showing that this technique may not be suitable to be used to pre-train large amounts of concatenated sensor data from different datasets.

Unlike the single dataset pre-training approach, which focuses on training the model to learn generic features from a single dataset, the N-1 implementation necessitates the model to possess the ability to recognize and differentiate both the source location and the dataset. This additional complexity adds a new dimension to the learning task, as the model must now extract relevant features for both aspects simultaneously. However, these features may be detrimental to the performance as the dataset used in the downstream task is unseen by the pre-trained model. Complex features extracted from the N-1 dataset may be irrelevant for the downstream task, leading the model to place emphasis on unnecessary information within the data when predicting activity labels.

The model's inability to perform HAR on unseen data may also mean that the extracted features from the pre-text task isn't generic in nature. The implementation with one dataset also has the benefit of allowing the model to train on the same data twice, as the model will have knowledge on the data from pre-training that can be utilized for fine-tuning. This can further be evidence that the pre-text task is better suited to extract dataset specific features. Another potential explanation for why the large pre-text dataset performs worse than the small pre-text dataset is that the large dataset contains data collected from different domains. Data collected from different experiments may have conflicting features, making it challenging for the model to learn coherent representations. Sensor data collected from different experiments can vary in terms of underlying patterns, noise levels, environmental conditions or other factors. These variations may have introduced inconsistencies or conflicts in the features present in the concatenated data. As a result, when the model attempts to learn coherent representations from such data, it may struggle to extract meaningful and generic features.

9.3 Impact of Using Variable Amounts of Data to Fine Tune Model

As discussed in the introduction, fully supervised models often suffer from lower performance when limited amount of data is available to train the model. The benefit of using a pre-text task is that the down stream task begins training with knowledge already gained. Given this, the performance of the final model should be high even when small amounts of data are available. The pre-trained model is compared to the fully-supervised model, where the number of users used for training is similarly limited.

Limiting the amount of data trained is done by training only a specific number of users from the training set. For example, if there are 10 users in the training set, only 3 users will be selected in order to train 30% of the dataset.

In the table below, we take motionsense as an example to look at the relationship between the number of users used for the downstream training and the reliability of the model. In both implementation, there is an expected general correlation between the number of users and the F1-score when downstream training. The model performs worse with a small amount of user data when the larger pre-training dataset is used. This is further evidence that the knowledge gained from the source task is not useful for the down stream task, and that the added complexity of the pre-trained model does not contribute to the down stream model's ability perform HAR.

The F-1 score in both implementations of motionsense slowly tapers off after 11 users have been trained. The diminishing improvement of the F1-score is also apparent in other datasets. This tapering off could indicate that there may be a saturation point in which the model has captured the most relevant information from the available user data, while further additions of user data have limited impact on improving the F1-score. Understanding this trend can help guide the decision-making process when determining the optimal number of users to include in the training process. Further investigation or experimentation should be conducted to explore the underlying reasons for the tapering off of the F1-score beyond a certain number of users. This understanding will help refine the training process and identify strategies to maximize the benefits derived from incorporating more user data while avoiding unnecessary computational or logistical burdens.

However, motionsense is the only dataset where the pre-trained model performed

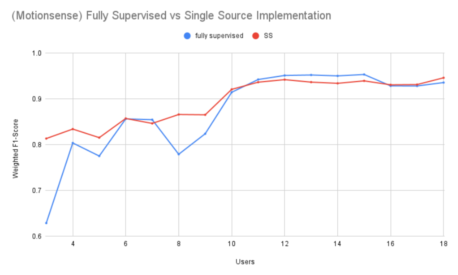


Figure 7: Comparing the fully supervised model vs proposed model

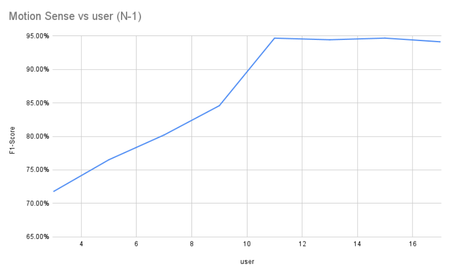


Figure 8: Motionsense vs user (N-1 implementation)

better at a low amount of users compared to their fully supervised counterpart. For example with HHAR, while the pre-trained model ultimately performs better when fine-tuned with 6 users, the fully-supervised model has a better F-1 score when fine-tuned with 3 users. This experiment shows that the proposed pre-training technique does not extract useful knowledge, as a fully-supervised model is consistently performing better without the pre-trained parameters.

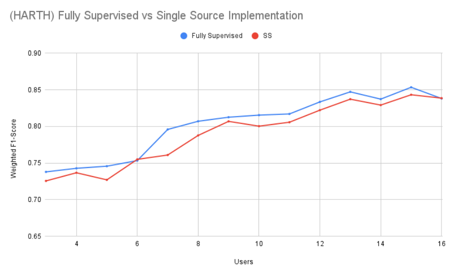


Figure 9: Comparing the fully supervised model vs proposed model (HARTH)

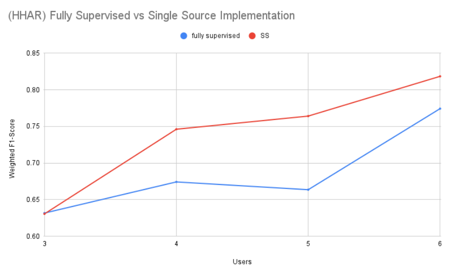


Figure 10: Comparing the fully supervised model vs proposed model (HHAR)

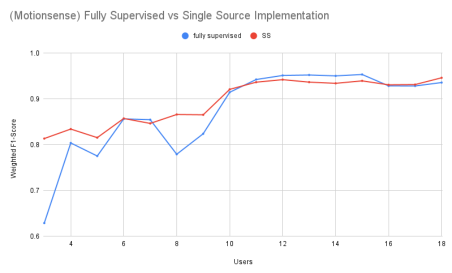


Figure 11: Comparing the fully supervised model vs proposed model (Motionsense)

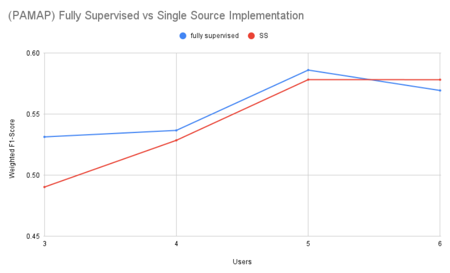


Figure 12: Comparing the fully supervised model vs proposed model (PAMAP)

10 Conclusions

This paper introduces a new pre-text task that uses the source location of the signal as a pseudo-label to pre-train a model used for HAR tasks. The task itself is straight forward to implement and can be implemented even with large amounts of unlabeled data. While the model showed mixed results in its performance, its performance on the HHAR dataset shows that the model has potential to extract insightful features through the pre-training process. However, the pre-trained model struggles to accurately categorize activities when fine-tuned with small amounts of data, even compared to the fully-supervised model. In addition, the pre-text task doesn't extract generic features when trained with a large and diverse dataset. The proposed model is only able to achieve good results when trained and fine-tuned on one dataset, as well as when the fine-tuning is done with an appropriately large dataset. This means that the proposed training task is not able to utilize large amounts of available unlabeled data to develop a HAR model.

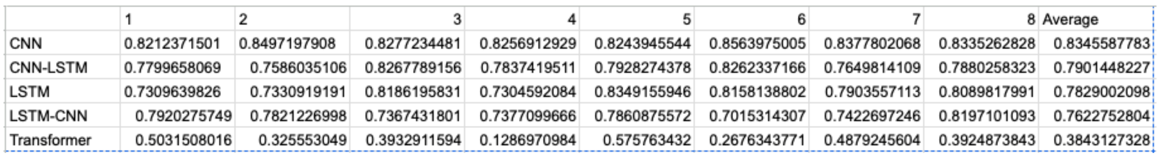
10.1 Further Investigation

While this project has focused on using one single modality, it would be interesting to see if the performance increases when combining models trained on multiple sensors. Additionally, it would be worthwhile to research the affects of concatenating multiple pre-trained tasks in improving performance.

A Appendix

A.1 Testing

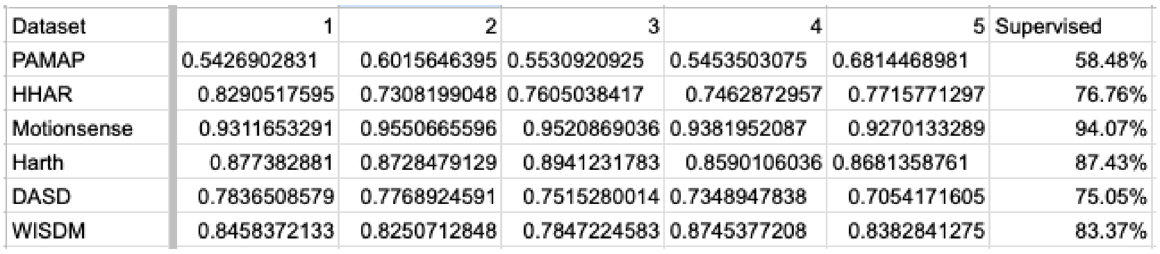
A.1.1 Choosing Between Architecture



	1	2	3	4	5	6	7	8	Average
CNN	0.8212371501	0.8497197908	0.8277234481	0.8256912929	0.8243945544	0.8563975005	0.8377802068	0.8335262828	0.8345587783
CNN-LSTM	0.7799658069	0.7586035106	0.8267789156	0.7837419511	0.7928274378	0.8262337166	0.7649814109	0.7880258323	0.7901448227
LSTM	0.7309639826	0.7330919191	0.8186195831	0.7304592084	0.8349155946	0.8158138802	0.7903557113	0.8089817991	0.7829002098
LSTM-CNN	0.7920275749	0.7821226998	0.7367431801	0.7377099666	0.7860875572	0.7015314307	0.7422697246	0.8197101093	0.7622752804
Transformer	0.5031508016	0.325553049	0.3932911594	0.1286970984	0.575763432	0.2676343771	0.4879245604	0.3924873843	0.3843127328

Figure 13: Weighed F1-Score of each architecture trained on HHAR

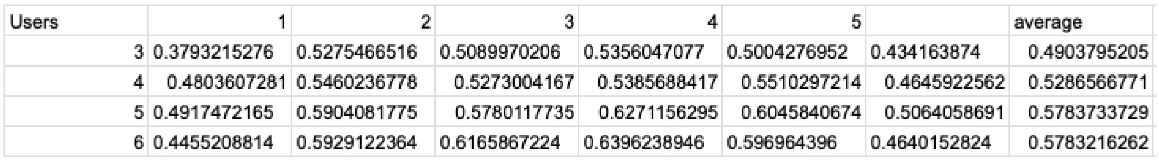
A.1.2 Raw data for fully supervised implementation



Dataset	1	2	3	4	5	Supervised
PAMAP	0.5426902831	0.6015646395	0.5530920925	0.5453503075	0.6814468981	58.48%
HHAR	0.8290517595	0.7308199048	0.7605038417	0.7462872957	0.7715771297	76.76%
Motionsense	0.9311653291	0.9550665596	0.9520869036	0.9381952087	0.9270133289	94.07%
Harth	0.877382881	0.8728479129	0.8941231783	0.8590106036	0.8681358761	87.43%
DASD	0.7836508579	0.7768924591	0.7515280014	0.7348947838	0.7054171605	75.05%
WISDM	0.8458372133	0.8250712848	0.7847224583	0.8745377208	0.8382841275	83.37%

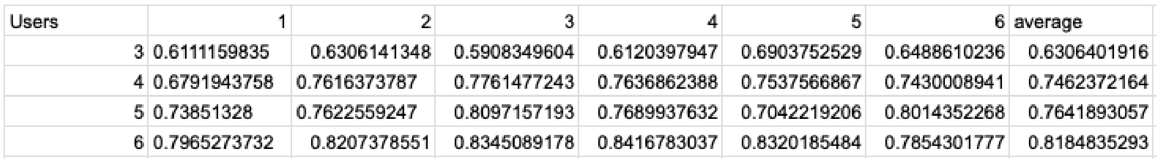
Figure 14: 5 runs of training fully supervised model will all user data

A.1.3 Raw Data



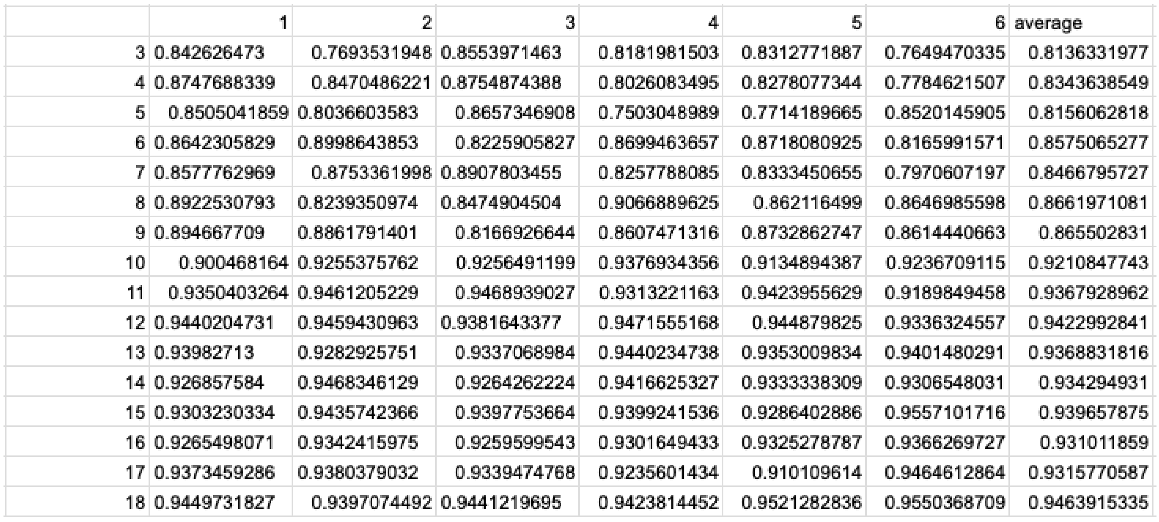
Users	1	2	3	4	5		average
3	0.3793215276	0.5275466516	0.5089970206	0.5356047077	0.5004276952	0.434163874	0.4903795205
4	0.4803607281	0.5460236778	0.5273004167	0.5385688417	0.5510297214	0.4645922562	0.5286566771
5	0.4917472165	0.5904081775	0.5780117735	0.6271156295	0.6045840674	0.5064058691	0.5783733729
6	0.4455208814	0.5929122364	0.6165867224	0.6396238946	0.596964396	0.4640152824	0.5783216262

Figure 15: PAMAP Raw Evaluation data for SS implementation



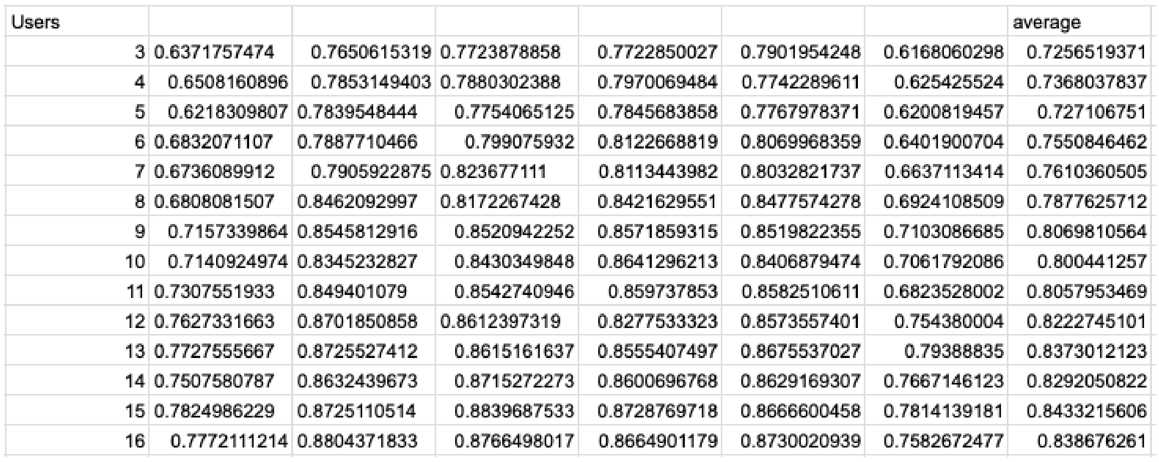
Users	1	2	3	4	5	6	average
3	0.6111159835	0.6306141348	0.5908349604	0.6120397947	0.6903752529	0.6488610236	0.6306401916
4	0.6791943758	0.7616373787	0.7761477243	0.7636862388	0.7537566867	0.7430008941	0.7462372164
5	0.73851328	0.7622559247	0.8097157193	0.7689937632	0.7042219206	0.8014352268	0.7641893057
6	0.7965273732	0.8207378551	0.8345089178	0.8416783037	0.8320185484	0.7854301777	0.8184835293

Figure 16: HHAR Raw Evaluation data for SS implementation



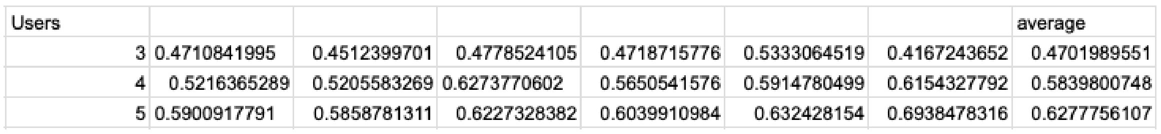
	1	2	3	4	5	6	average
3	0.842626473	0.7693531948	0.8553971463	0.8181981503	0.8312771887	0.7649470335	0.8136331977
4	0.8747688339	0.8470486221	0.8754874388	0.8026083495	0.8278077344	0.7784621507	0.8343638549
5	0.8505041859	0.8036603583	0.8657346908	0.7503048989	0.7714189665	0.8520145905	0.8156062818
6	0.8642305829	0.8998643853	0.8225905827	0.8699463657	0.8718080925	0.8165991571	0.8575065277
7	0.8577762969	0.8753361998	0.8907803455	0.8257788085	0.8333450655	0.7970607197	0.8466795727
8	0.8922530793	0.8239350974	0.8474904504	0.9066889625	0.862116499	0.8646985598	0.8661971081
9	0.894667709	0.8861791401	0.8166926644	0.8607471316	0.8732862747	0.8614440663	0.865502831
10	0.900468164	0.9255375762	0.9256491199	0.9376934356	0.9134894387	0.9236709115	0.9210847743
11	0.9350403264	0.9461205229	0.9468939027	0.9313221163	0.9423955629	0.9189849458	0.9367928962
12	0.9440204731	0.9459430963	0.9381643377	0.9471555168	0.944879825	0.9336324557	0.9422992841
13	0.93982713	0.9282925751	0.9337068984	0.9440234738	0.9353009834	0.9401480291	0.9368831816
14	0.926857584	0.9468346129	0.9264262224	0.9416625327	0.9333338309	0.9306548031	0.934294931
15	0.9303230334	0.9435742366	0.9397753664	0.9399241536	0.9286402886	0.9557101716	0.939657875
16	0.9265498071	0.9342415975	0.9259599543	0.9301649433	0.9325278787	0.9366269727	0.931011859
17	0.9373459286	0.9380379032	0.9339474768	0.9235601434	0.910109614	0.9464612864	0.9315770587
18	0.9449731827	0.9397074492	0.9441219695	0.9423814452	0.9521282836	0.9550368709	0.9463915335

Figure 17: Motionsense Raw Evaluation data for SS implementation



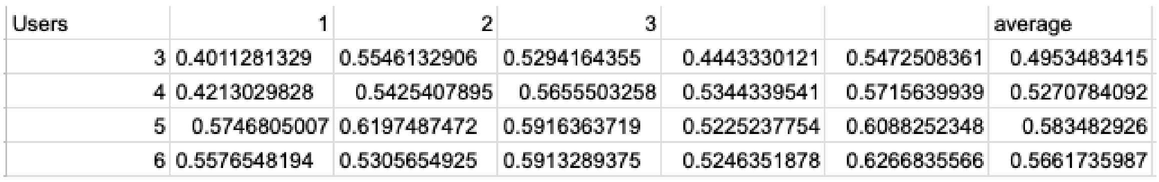
Users							average
3	0.6371757474	0.7650615319	0.7723878858	0.7722850027	0.7901954248	0.6168060298	0.7256519371
4	0.6508160896	0.7853149403	0.7880302388	0.7970069484	0.7742289611	0.625425524	0.7368037837
5	0.6218309807	0.7839548444	0.7754065125	0.7845683858	0.7767978371	0.6200819457	0.727106751
6	0.6832071107	0.7887710466	0.799075932	0.8122668819	0.8069968359	0.6401900704	0.7550846462
7	0.6736089912	0.7905922875	0.823677111	0.8113443982	0.8032821737	0.6637113414	0.7610360505
8	0.6808081507	0.8462092997	0.8172267428	0.8421629551	0.8477574278	0.6924108509	0.7877625712
9	0.7157339864	0.8545812916	0.8520942252	0.8571859315	0.8519822355	0.7103086685	0.8069810564
10	0.7140924974	0.8345232827	0.8430349848	0.8641296213	0.8406879474	0.7061792086	0.800441257
11	0.7307551933	0.849401079	0.8542740946	0.859737853	0.8582510611	0.6823528002	0.8057953469
12	0.7627331663	0.8701850858	0.8612397319	0.8277533323	0.8573557401	0.754380004	0.8222745101
13	0.7727555667	0.8725527412	0.8615161637	0.8555407497	0.8675537027	0.79388835	0.8373012123
14	0.7507580787	0.8632439673	0.8715272273	0.8600696768	0.8629169307	0.7667146123	0.8292050822
15	0.7824986229	0.8725110514	0.8839687533	0.8728769718	0.8666600458	0.7814139181	0.8433215606
16	0.7772111214	0.8804371833	0.8766498017	0.8664901179	0.8730020939	0.7582672477	0.838676261

Figure 18: HARTH Raw Evaluation data for SS implementation



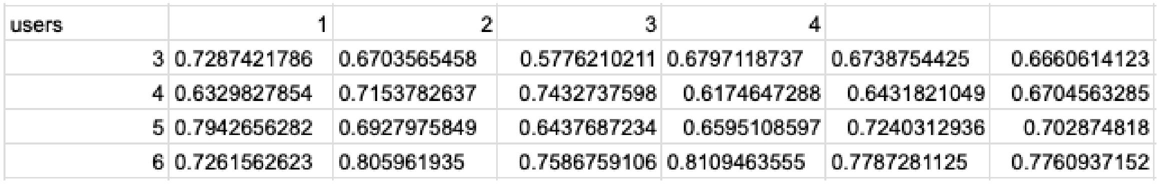
Users							average
3	0.4710841995	0.4512399701	0.4778524105	0.4718715776	0.5333064519	0.4167243652	0.4701989551
4	0.5216365289	0.5205583269	0.6273770602	0.5650541576	0.5914780499	0.6154327792	0.5839800748
5	0.5900917791	0.5858781311	0.6227328382	0.6039910984	0.632428154	0.6938478316	0.6277756107

Figure 19: DASD



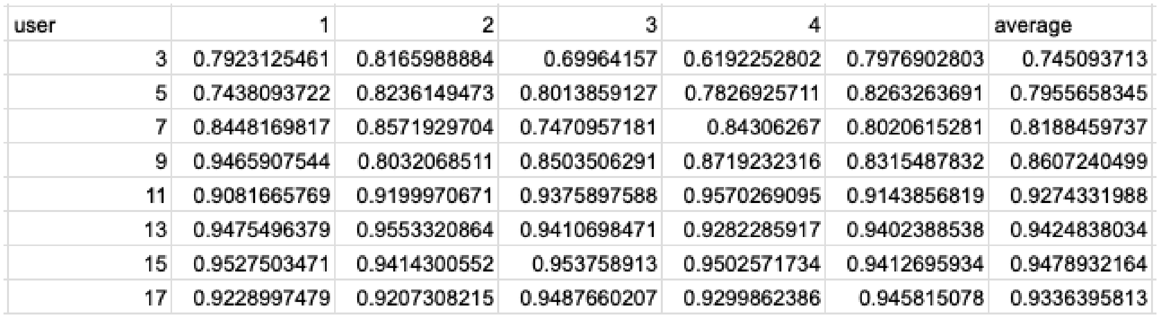
Users	1	2	3			average
3	0.4011281329	0.5546132906	0.5294164355	0.4443330121	0.5472508361	0.4953483415
4	0.4213029828	0.5425407895	0.5655503258	0.5344339541	0.5715639939	0.5270784092
5	0.5746805007	0.6197487472	0.5916363719	0.5225237754	0.6088252348	0.583482926
6	0.5576548194	0.5305654925	0.5913289375	0.5246351878	0.6266835566	0.5661735987

Figure 20: PAMAP Raw Evaluation Data N-1 Implementation



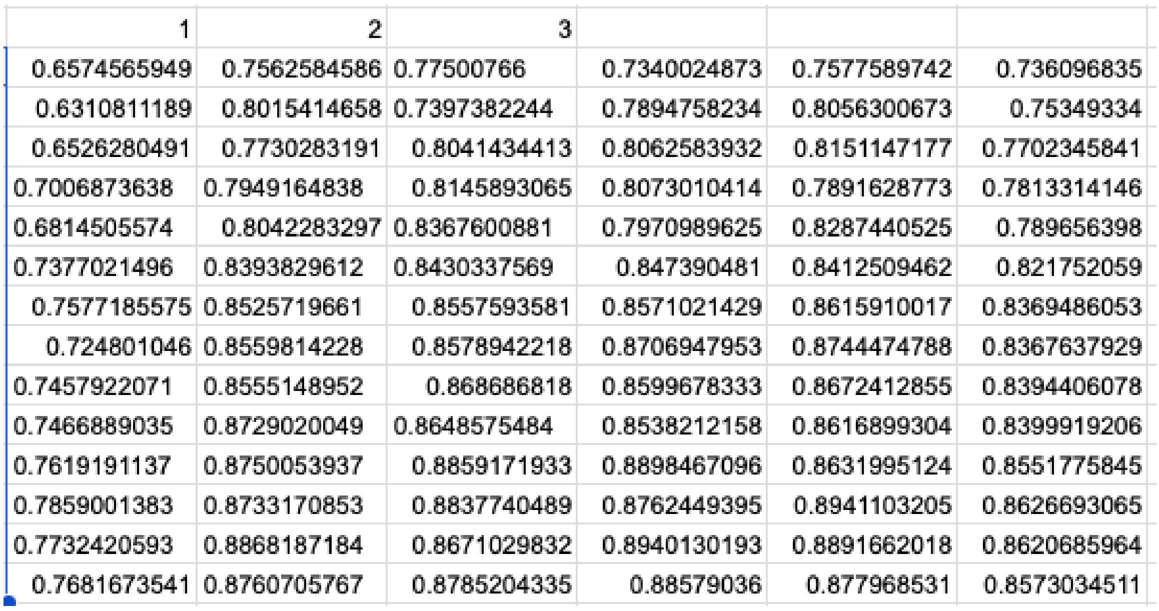
users	1	2	3	4		
3	0.7287421786	0.6703565458	0.5776210211	0.6797118737	0.6738754425	0.6660614123
4	0.6329827854	0.7153782637	0.7432737598	0.6174647288	0.6431821049	0.6704563285
5	0.7942656282	0.6927975849	0.6437687234	0.6595108597	0.7240312936	0.702874818
6	0.7261562623	0.805961935	0.7586759106	0.8109463555	0.7787281125	0.7760937152

Figure 21: HHAR Raw Evaluation Data N-1 Implementation



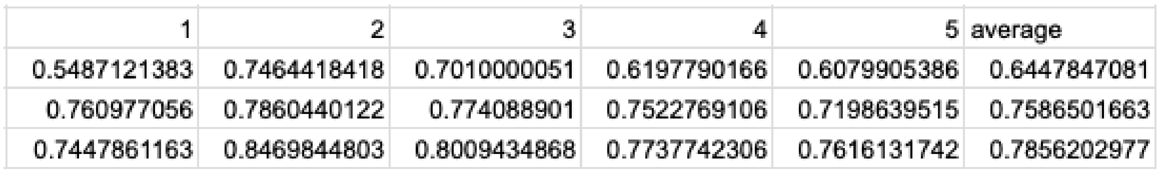
user	1	2	3	4		average
3	0.7923125461	0.8165988884	0.69964157	0.6192252802	0.7976902803	0.745093713
5	0.7438093722	0.8236149473	0.8013859127	0.7826925711	0.8263263691	0.7955658345
7	0.8448169817	0.8571929704	0.7470957181	0.84306267	0.8020615281	0.8188459737
9	0.9465907544	0.8032068511	0.8503506291	0.8719232316	0.8315487832	0.8607240499
11	0.9081665769	0.9199970671	0.9375897588	0.9570269095	0.9143856819	0.9274331988
13	0.9475496379	0.9553320864	0.9410698471	0.9282285917	0.9402388538	0.9424838034
15	0.9527503471	0.9414300552	0.953758913	0.9502571734	0.9412695934	0.9478932164
17	0.9228997479	0.9207308215	0.9487660207	0.9299862386	0.945815078	0.9336395813

Figure 22: Motionsense Raw Evaluation Data N-1 Implementation



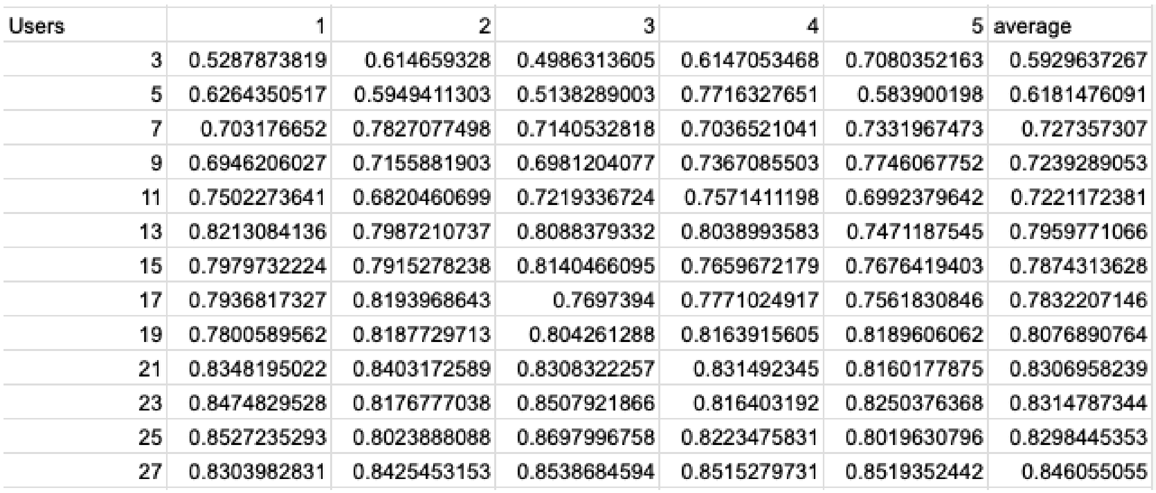
1	2	3			
0.6574565949	0.7562584586	0.77500766	0.7340024873	0.7577589742	0.736096835
0.6310811189	0.8015414658	0.7397382244	0.7894758234	0.8056300673	0.75349334
0.6526280491	0.7730283191	0.8041434413	0.8062583932	0.8151147177	0.7702345841
0.7006873638	0.7949164838	0.8145893065	0.8073010414	0.7891628773	0.7813314146
0.6814505574	0.8042283297	0.8367600881	0.7970989625	0.8287440525	0.789656398
0.7377021496	0.8393829612	0.8430337569	0.847390481	0.8412509462	0.821752059
0.7577185575	0.8525719661	0.8557593581	0.8571021429	0.8615910017	0.8369486053
0.724801046	0.8559814228	0.8578942218	0.8706947953	0.8744474788	0.8367637929
0.7457922071	0.8555148952	0.868686818	0.8599678333	0.8672412855	0.8394406078
0.7466889035	0.8729020049	0.8648575484	0.8538212158	0.8616899304	0.8399919206
0.7619191137	0.8750053937	0.8859171933	0.8898467096	0.8631995124	0.8551775845
0.7859001383	0.8733170853	0.8837740489	0.8762449395	0.8941103205	0.8626693065
0.7732420593	0.8868187184	0.8671029832	0.8940130193	0.8891662018	0.8620685964
0.7681673541	0.8760705767	0.8785204335	0.88579036	0.877968531	0.8573034511

Figure 23: HARTH Raw Evaluation Data N-1 Implementation



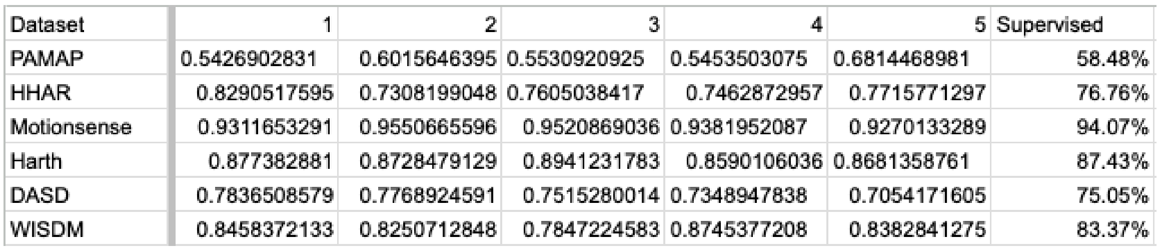
1	2	3	4	5	average
0.5487121383	0.7464418418	0.7010000051	0.6197790166	0.6079905386	0.6447847081
0.760977056	0.7860440122	0.774088901	0.7522769106	0.7198639515	0.7586501663
0.7447861163	0.8469844803	0.8009434868	0.7737742306	0.7616131742	0.7856202977

Figure 24: DASD Raw Evaluation Data N-1 Implementation



Users	1	2	3	4	5	average
3	0.5287873819	0.614659328	0.4986313605	0.6147053468	0.7080352163	0.5929637267
5	0.6264350517	0.5949411303	0.5138289003	0.7716327651	0.583900198	0.6181476091
7	0.703176652	0.7827077498	0.7140532818	0.7036521041	0.7331967473	0.727357307
9	0.6946206027	0.7155881903	0.6981204077	0.7367085503	0.7746067752	0.7239289053
11	0.7502273641	0.6820460699	0.7219336724	0.7571411198	0.6992379642	0.7221172381
13	0.8213084136	0.7987210737	0.8088379332	0.8038993583	0.7471187545	0.7959771066
15	0.7979732224	0.7915278238	0.8140466095	0.7659672179	0.7676419403	0.7874313628
17	0.7936817327	0.8193968643	0.7697394	0.7771024917	0.7561830846	0.7832207146
19	0.7800589562	0.8187729713	0.804261288	0.8163915605	0.8189606062	0.8076890764
21	0.8348195022	0.8403172589	0.8308322257	0.831492345	0.8160177875	0.8306958239
23	0.8474829528	0.8176777038	0.8507921866	0.816403192	0.8250376368	0.8314787344
25	0.8527235293	0.8023888088	0.8697996758	0.8223475831	0.8019630796	0.8298445353
27	0.8303982831	0.8425453153	0.8538684594	0.8515279731	0.8519352442	0.846055055

Figure 25: WISDM Raw Evaluation Data N-1 Implementation



Dataset	1	2	3	4	5	Supervised
PAMAP	0.5426902831	0.6015646395	0.5530920925	0.5453503075	0.6814468981	58.48%
HHAR	0.8290517595	0.7308199048	0.7605038417	0.7462872957	0.7715771297	76.76%
Motionsense	0.9311653291	0.9550665596	0.9520869036	0.9381952087	0.9270133289	94.07%
Harth	0.877382881	0.8728479129	0.8941231783	0.8590106036	0.8681358761	87.43%
DASD	0.7836508579	0.7768924591	0.7515280014	0.7348947838	0.7054171605	75.05%
WISDM	0.8458372133	0.8250712848	0.7847224583	0.8745377208	0.8382841275	83.37%

Figure 26: Fully Supervised Model trained with all available Users

A.2 User Manual

To run the evaluation, first you will need to collect data from the sources. For fast raw data pre-processing, the dataset folders can be structured as seen in Figure 27. If the dataset folders are done correctly, the script pickle_files.py can be run and the data from all the datasets will pickled into a separate folder. From there on, notebooks and scripts can be run to evaluate the model. The pickle_files.py script can be edited to correctly pickle from the correct folder.

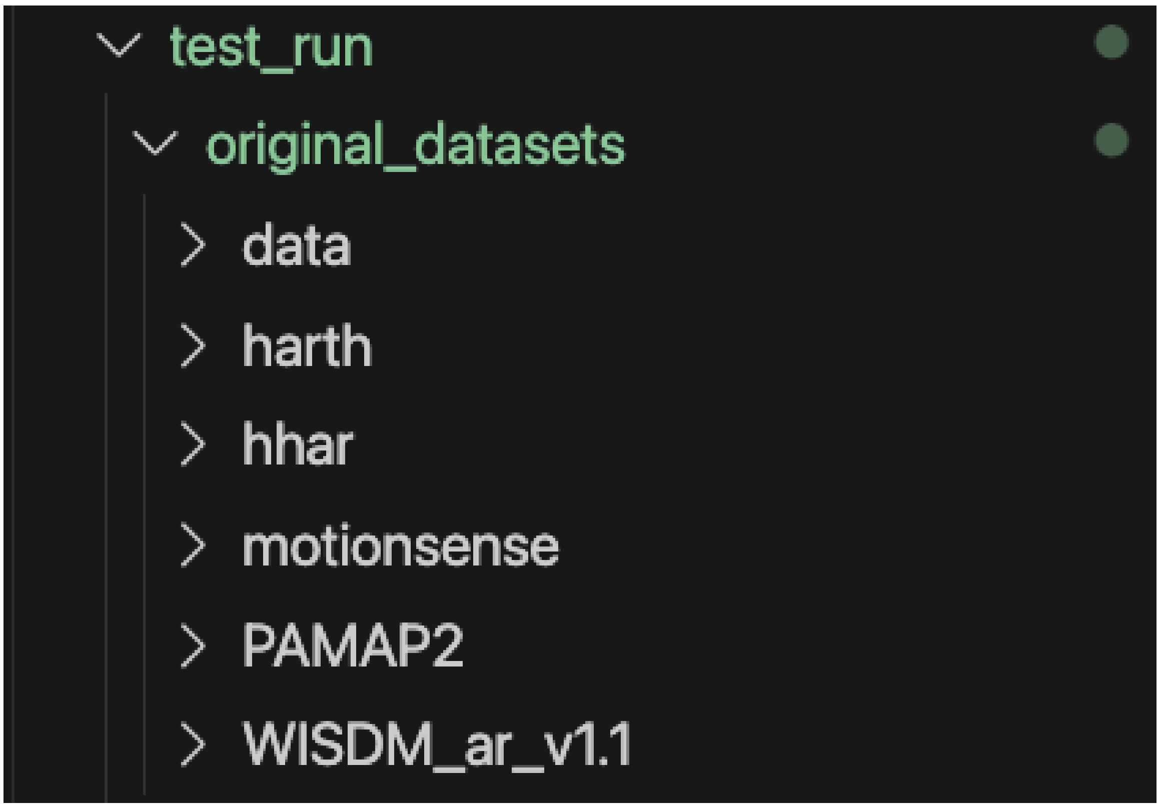


Figure 27: Raw Data Folder Structure

The scripts and notebooks use the pickled datasets as it saves time having to pre-process the data repeatedly.

Notebook Name	Description
eval_all_datasets	Evaluates the SS implementation
Eval_n-1	Evaluates the N-1 implementation
Fully-Supervised	Evaluates the fully supervised implementation

Table 3: Notebook Description

A.3 Other Appendices

A.3.1 Additional Confusion Matrix

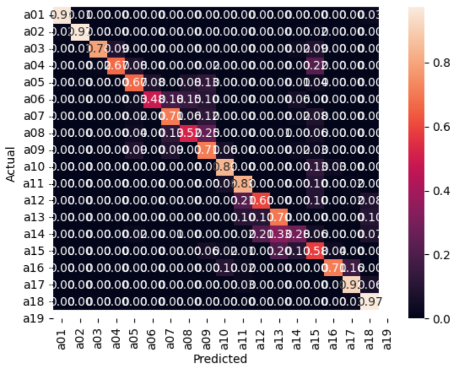


Figure 28: Confusion Matrix of DASD (SS)

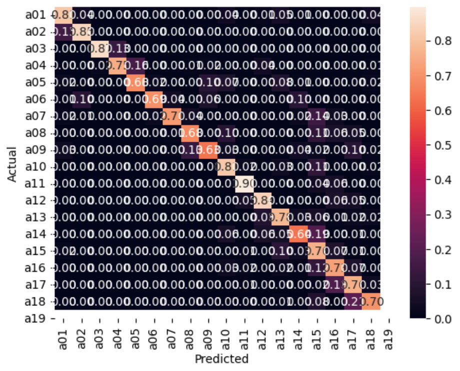


Figure 29: Confusion Matrix of DASD (N-1)

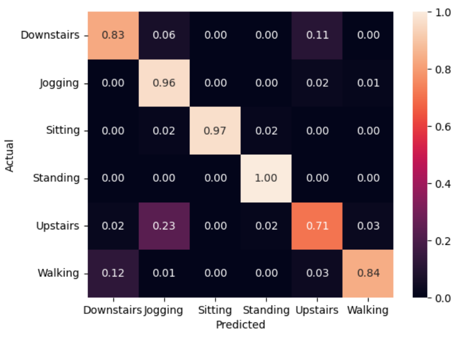


Figure 30: Confusion Matrix of WISDM (N-1)

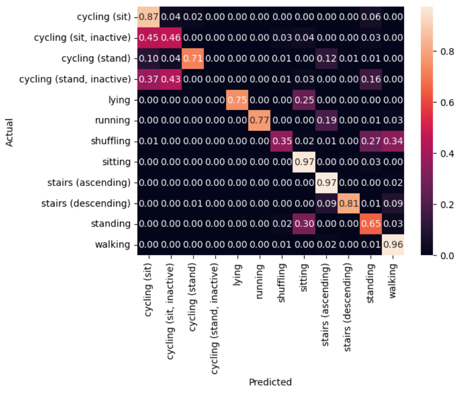


Figure 31: Confusion Matrix of HARTH (SS)

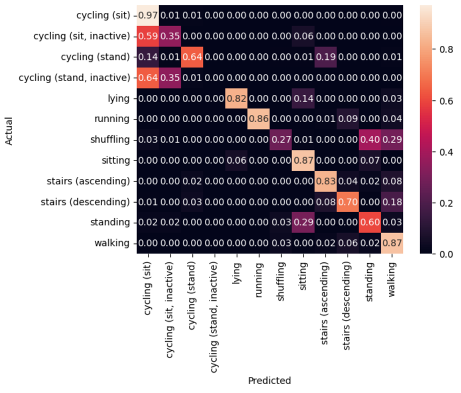


Figure 32: Confusion Matrix of HARTH (N-1)

References

[1] Billur Barshan and Kerem Altun. Daily and Sports Activities. UCI Machine Learning Repository, 2013. DOI: https://doi.org/10.24432/C5C59F.

[2] Yoshua Bengio, Ian Goodfellow, and Aaron Courville. *Deep learning*, volume 1. MIT press Cambridge, MA, USA, 2017.

[3] Bhattacharya Sourav Prentow Thor Kjrgaard Mikkel Blunck, Henrik and Anind Dey. Heterogeneity Activity Recognition. UCI Machine Learning Repository, 2015. DOI: https://doi.org/10.24432/C5689X.

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

[5] Sravan Kumar Challa, Akhilesh Kumar, and Vijay Bhaskar Semwal. A multi-branch cnn-bilstm model for human activity recognition using wearable sensor data. *The Visual Computer*, 38(12):4095–4109, 2022.

[6] Diane Cook, Kyle D Feuz, and Narayanan C Krishnan. Transfer learning for activity recognition: A survey. *Knowledge and information systems*, 36:537–556, 2013.

[7] Iveta Dirgová Luptáková, Martin Kubovčík, and Jiří Pospíchal. Wearable sensor-based human activity recognition with transformer model. *Sensors*, 22(5):1911, 2022.

[8] Harish Haresamudram, Irfan Essa, and Thomas Plötz. Assessing the state of self-supervised human activity recognition using wearables. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 6(3), sep 2022.

[9] Sen He and Nicolas Pugeault. Deep saliency: What is learnt by a deep network about saliency? *arXiv preprint arXiv:1801.04261*, 2018.

[10] Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. A survey on contrastive self-supervised learning. *Technologies*, 9(1):2, 2020.

[11] Ken J. Kubota, Jason A. Chen, and Max A. Little. Machine learning for large-scale wearable sensor data in parkinson's disease: Concepts, promises, pitfalls, and futures. *Movement Disorders*, 31(9):1314–1326, 2016.

[12] Jennifer R Kwapisz, Gary M Weiss, and Samuel A Moore. Activity recognition using cell phone accelerometers. *ACM SigKDD Explorations Newsletter*, 12(2):74–82, 2011.

[13] Oscar D. Lara and Miguel A. Labrador. A survey on human activity recognition using wearable sensors. *IEEE Communications Surveys Tutorials*, 15(3):1192–1209, Third 2013.

[14] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[15] Kenneth Leung. Micro, macro weighted averages of f1 score, clearly explained, Sep 2022.

[16] Kongsvold Atle Bach Kerstin Bårdstu Hilde Bremseth Logacjov, Aleksej and Paul Jarle Mork. HARTH. UCI Machine Learning Repository, 2023. DOI: https://doi.org/10.24432/C5NC90.

[17] Wenjie Lu, Jiazheng Li, Yifan Li, Aijun Sun, and Jingyang Wang. A cnn-lstm-based model to forecast stock prices. *Complexity*, 2020:1–10, 2020.

[18] Mohammad Malekzadeh, Richard G. Clegg, Andrea Cavallaro, and Hamed Haddadi. Mobile sensor data anonymization. In *Proceedings of the International Conference on Internet of Things Design and Implementation*, IoTDI '19, pages 49–58, New York, NY, USA, 2019. ACM.

[19] L. Minh Dang, Kyungbok Min, Hanxiang Wang, Md. Jalil Piran, Cheol Hee Lee, and Hyeonjoon Moon. Sensor-based and vision-based human activity recognition: A comprehensive survey. *Pattern Recognition*, 108:107561, 2020.

[20] Theodoros Ntakouris. Keras documentation: Timeseries classification with a transformer model, Jun 2021.

[21] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[22] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[23] Sreenivasan Ramasamy Ramamurthy and Nirmalya Roy. Recent trends in machine learning for human activity recognition—a survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1254, 2018.

[24] YCAP Reddy, P Viswanath, and B Eswara Reddy. Semi-supervised learning: A brief review. *Int. J. Eng. Technol*, 7(1.8):81, 2018.

[25] Attila Reiss. PAMAP2 Physical Activity Monitoring. UCI Machine Learning Repository, 2012. DOI: https://doi.org/10.24432/C5NW2H.

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

[27] Jubil T Sunny, Sonia Mary George, Jubilant J Kizhakkethottam, Jubil T Sunny, Sonia Mary George, and Jubilant J Kizhakkethottam. Applications and challenges of human activity recognition using sensors in a smart environment. *IJIRST Int. J. Innov. Res. Sci. Technol*, 2:50–57, 2015.

[28] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part III 27*, pages 270–279. Springer, 2018.

[29] Chi Ian Tang, Ignacio Perez-Pozuelo, Dimitris Spathis, Soren Brage, Nick Wareham, and Cecilia Mascolo. Selfhar: Improving human activity recognition through self-training with unlabeled data. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 5(1), mar 2021.

[30] Chao Tao, Ji Qi, Mingning Guo, Qing Zhu, and Haifeng Li. Self-supervised remote sensing feature learning: Learning paradigms, challenges, and future works. *IEEE Transactions on Geoscience and Remote Sensing*, 2023.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[32] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, Eftychios Protopapadakis, et al. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018, 2018.

[33] Michalis Vrigkas, Christophoros Nikou, and Ioannis A. Kakadiaris. A review of human activity recognition methods. *Frontiers in Robotics and AI*, 2, 2015.

[34] Lirong Wu, Haitao Lin, Cheng Tan, Zhangyang Gao, and Stan Z. Li. Self-supervised learning on graphs: Contrastive, generative, or predictive. *IEEE Transactions on Knowledge and Data Engineering*, 35(4):4216–4235, 2023.

[35] Kun Xia, Jianguang Huang, and Hanyu Wang. Lstm-cnn architecture for human activity recognition. *IEEE Access*, 8:56855–56866, 2020.

[36] Yaochen Xie, Zhao Xu, Jingtun Zhang, Zhengyang Wang, and Shuiwang Ji. Self-supervised learning of graph neural networks: A unified review. *IEEE transactions on pattern analysis and machine intelligence*, 45(2):2412–2429, 2022.

[37] Jian Bo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiao Li Li, and Shonali Krishnaswamy. Deep convolutional neural networks on multichannel time series for human activity recognition. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 3995–4001. AAAI Press, 2015.

[38] Saeedeh Zebhi, S. M. T. AlModarresi, and Vahid Abootalebi. Human activity recognition using pre-trained network with informative templates. *International Journal of Machine Learning and Cybernetics*, 12(12):3449–3461, Dec 2021.

[39] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. S4l: Self-supervised semi-supervised learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1476–1485, 2019.

[40] Yu Zhao, Rennong Yang, Guillaume Chevalier, Ximeng Xu, and Zhenxing Zhang. Deep residual bidir-lstm for human activity recognition using wearable sensors. *Mathematical Problems in Engineering*, 2018:1–13, 2018.

[41] Zehui Zhao, Laith Alzubaidi, Jinglan Zhang, Ye Duan, and Yuantong Gu. A comparison review of transfer learning and self-supervised learning: Definitions, applications, advantages and limitations. *Expert Systems with Applications*, 242:122807, 2024.