NANYANG TECHNOLOGICAL UNIVERSITY

Deep CNN-LSTM Supervised model and CNN Self-supervised model for Human Activity Recognition

PSCSE21-0033

Submitted in Partial Fulfillment of the Requirements for the Degree of Bachelor of Computer Science of the Nanyang Technological University

by

Liao Zixin
U1920313L

April 14, 2023

Supervised by Associate Professor Kwoh Chee Keong

ABSTRACT

Human Activity Recognition (HAR) has garnered significant interest from researchers in past decades. With the quick development of wearable sensor technology and the high availability of smart devices, e.g., accelerometers and gyroscopes embedded in smartphones, HAR has become a popular field of research recently. In this paper, we propose a framework for HAR data classification which automatically extracts spatial and temporal features from smart device sensory data. This is achieved via a hybrid supervised learning architecture, that consists of a Convolutional Neural Network (CNN), and a Long Short-Term Memory Network (LSTM). However, a large amount of labeled data is typically required to perform supervised learning, which can be lacking due to data privacy concerns and the high cost of manual labeling in real-world scenarios. Therefore, learning from the large amounts of unlabeled data becomes crucial. To this end, we propose a self-supervised learning (SSL) framework that learns useful representations from unlabeled HAR sensory data. Our framework consists of two stages: 1) self-supervised pretraining, where we propose a set of pretext tasks to help the model learn from unlabeled data, and 2) fine-tuning the pre-trained model with the few available labeled samples according to the original HAR task. The results demonstrate that our SSL approach significantly improves the model performance compared to supervised training given limited labeled samples. In addition, by fine-tuning the simple 1-D CNN pre-trained self-supervised model using only **5%** of labeled data, we can attain a level of performance that is comparable to the complex CNN-LSTM supervised training with full labels. Last, we observe that self-supervised pre-training assists the models in developing robustness to data imbalanced issues. The source code is available on `https://github.com/LizLicense/HAR-CNN-LSTM-ATT-pyTorch.git`.

Keywords— Deep learning, Supervised learning, Self-Supervised Learning, Human Activity Recognition, Robustness

CONTENTS

1 Introduction **4**

2 Related work **6**

2.1 Machine Learning-based HAR Systems 6

2.2 Deep learning-based HAR Systems 6

2.3 Self-supervised Learning 6

3 Proposed Methodology **7**

3.1 List of Symbols 7

3.2 Supervised Learning Framework 7

3.2.1 Overview 7

3.2.2 Proposed CNN-LSTM-Attention Network 7

3.3 Self-supervised Learning 9

3.3.1 Overview 9

3.3.2 Stage 1: Pretext Task 11

3.3.3 Stage 2: Downstream Task - Learning to Classify Activity 11

4 Experimental Results **12**

4.1 Dataset summary 12

4.2 Proposed Methods Results 12
4.2.1 Supervised Model Results 12
4.2.2 Self-supervised Learning Results 13
4.3 HAPT Data Imbalance Problem 15
4.4 Which sensor is more efficient? 17

5 Conclusion **18**

6 Acknowledgement **18**

A Dataset **21**

A.1 UCIHAR Dataset 21
A.2 HAPT Dataset 22
A.3 HHAR Dataset 23
A.3.1 HHAR Dataset - Phone 24
A.3.2 HHAR Dataset - Watch 24
A.4 Data preprocess 24

B Network hyperparameters **26**

C Material Resources and Costing **27**

D Project planning **27**

List of Figures

1 An overview of the proposed SSL. 5
2 Architecture of CNN-LSTM-Attention network. 8
3 Detail architecture of CNN-LSTM-Attention network. 8
4 Architecture of self-supervised learning network. 10
5 CNN-LSTM network train and test accuracy on three datasets. 13
6 Fine-tuning the pretrained SSL networks on three dataset - f1-score(%). 14
7 Fine-tuning the pretrained SSL networks on three dataset - total loss. 14
8 HAPT activity counts before and after replication. 16
9 HAPT(imbalanced) confusion matrix. 16
10 UCIHAR data count per activity. 22
11 UCIHAR data provided by each user. 22
12 HAPT(imbalanced) data count per activity. 23
13 HAPT(imbalanced) data provided by each user. 24
14 HHAR phone data provided by each user. 25
15 HHAR watch data provided by each user. 26
16 Project planning. 28

List of Tables

1 Comparison of models on three Datasets using smartphone data, specifically accelerometer and gyroscope data. Replication is applied to the HAPT dataset. 12

2 Proposed supervised CNN-LSTM network comparison with baselines. 13

3 Comparison of the SSL performance on three datasets. Sup. stands for supervised CNN-LSTM training performance, while FT. refers to fine-tuning performance. 14

4 Baselines and comparison on UCIHAR dataset. 14

5 Comparison of the SSL performance with MSE loss and KLD loss. 15

6 Comparison of our proposed models with different methods to address the data imbalance problem on HAPT dataset. 17

7 Comparison of the SSL performance on HAPT dataset without oversampling. 17

8 Comparison of HHAR dataset with different devices and different sensors. 17

9 Comparison on different users of models on HHAR Dataset from smartwatch-gyroscope data. Variance equals the difference between Phone and Watch. 18

10 A brief description of three datasets. 21

11 Count of activity of HHAR dataset-smartphone. 25

12 Count of activity of HHAR dataset-smartwatch. 25

13 Supervised learning network hyperparameters set up on three datasets. 26

14 SSL network hyperparameters set up. 27

1 INTRODUCTION

The past decade witnessed a remarkable improvement in microelectronics and computer systems, permitting sensors and mobile devices with unparalleled characteristics. Human activity recognition (HAR) technology has obtained a growing interest in recent years. HAR system based on wearable sensors helps to provide valuable insights into human behavior, including temperature, heart rate, brain activity, and other critical areas (Edwards, 2012). Among the wearable devices used in HAR systems, smartphones, and smartwatches are becoming increasingly popular due to their portability and ubiquity. These devices are equipped with orientation and motion sensors such as accelerometers and gyroscopes, which can effectively categorize people's actions (Alrazzak & Alhalabi, 2019). Previous research has indicated that accelerometers and gyroscopes can be used to recognize common human activities and their use in HAR systems is gaining prominence(Lockhart et al., 2011).

Previously, numerous machine learning algorithms have been deployed to predict human activities with smart device sensors. Some machine learning models have yielded positive HAR outcomes. Anguita et al. (2012) achieved 89.3% precision with Support Vector Machines; Polu & Polu (2018) achieved 93.69% precision with Random Forest Algorithm; Ronao & Cho (2014) achieved 91.76% precision with Hidden Markov Model regression; Attal et al. (2015) achieved 94.62% with k-Nearest Neighbors. These approaches typically involve manual data feature extraction, which heavily relies on the expertise of domain experts. Even though the autoencoder enables extracting features automation in Kolosnjaji & Eckert (2015)'s work, supervised machine learning requires a vast amount of label data. Another weakness of machine learning, mentioned by Li et al. (2019), is its inability to handle large and complex datasets with high-dimensional input spaces. As the dimensionality of the data increases, traditional machine learning algorithms become less effective at finding the underlying patterns and relationships among the data.

On the other hand, deep learning approaches can automatically extract spatial and/or temporal features from input data without domain knowledge, which allows quick and improved progress in the field (Alzubaidi et al. (2021)). The neural network models gaining popularity, such as Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN), and Artificial Neural Networks(ANN). They can acquire semantic and emotional information from the input as well as capture temporal and spatial information. The traditional neural network models give rise to deep learning. In addition to being a network with multiple layers, it emphasizes the extraction of hidden and abstract information at higher levels.

The growing usage of smart devices has enabled the collection of large volumes of human activity datasets, which has played a significant role in the success of studies focused on HAR, for example, the works of Singh et al. (2020), Gupta (2021), and Gochoo et al. (2018) have all benefited from these abundant datasets. However, the success of these studies is highly dependent on the significant amount of labeled data in deep learning model training. In fact, this can be challenging in the case of smart devices, which continuously collect data on days and nights. It requires experts' domain knowledge to recognize and label sufficient human activity, making it difficult to collect enough labeled data. Moreover, many users are hesitant to share their health data with health labs due to data privacy concerns, further exacerbating the data shortage.

To address the above issue, self-supervised(SSL) learning has gained popularity in recent years due to the availability of large unlabeled datasets and the success of deep learning models. Which can be beneficial in situations where labeled data is scarce or expensive to obtain. By applying various transformations to the raw data, data augmentation techniques artificially increase the features of the training dataset. These techniques apply various transformations such as rotation, scaling, and jittering (Um et al., 2017), to the existing data to generate new and synthetic examples. It helps to reduce overfitting and improve the generalization of the model. Additionally, data augmentation techniques can also be used to expand the diversity of the training dataset and make the model more robust to variations in the data. In this paper, we apply four classifying augmentation techniques to three public HAR datasets with a novel self-supervised approach, which has rarely been discussed. Inspired by Xie et al. (2020)'s work, we are keen to leverage the loss function to improve training consistency and model robustness. Our method utilizes one-dimensional convolutional neural networks(1D-CNN) to extract meaningful features from users' physiological signals.

In this paper, we implement a 1D-CNN self-supervised network(Figure 1) and a CNN-LSTM-ATT fully supervised network(Figure 2). The fine-tuning self-supervised network performance with different data percentages is compared with the fully supervised model training with full labels. The experiment result is evaluated in test accuracy% and f1-score%.

The contributions of the paper are:

1. We developed an efficient CNN-LSTM-Attention network to classify HAR sensory data given a fully labeled dataset. The experimental results conducted on three public HAR datasets prove the efficiency of our framework.
2. From the hardware sensory perspective, we compared the fully supervised network performance of smartphones and smartwatches for human activity recognition using the same gyroscope and ac-

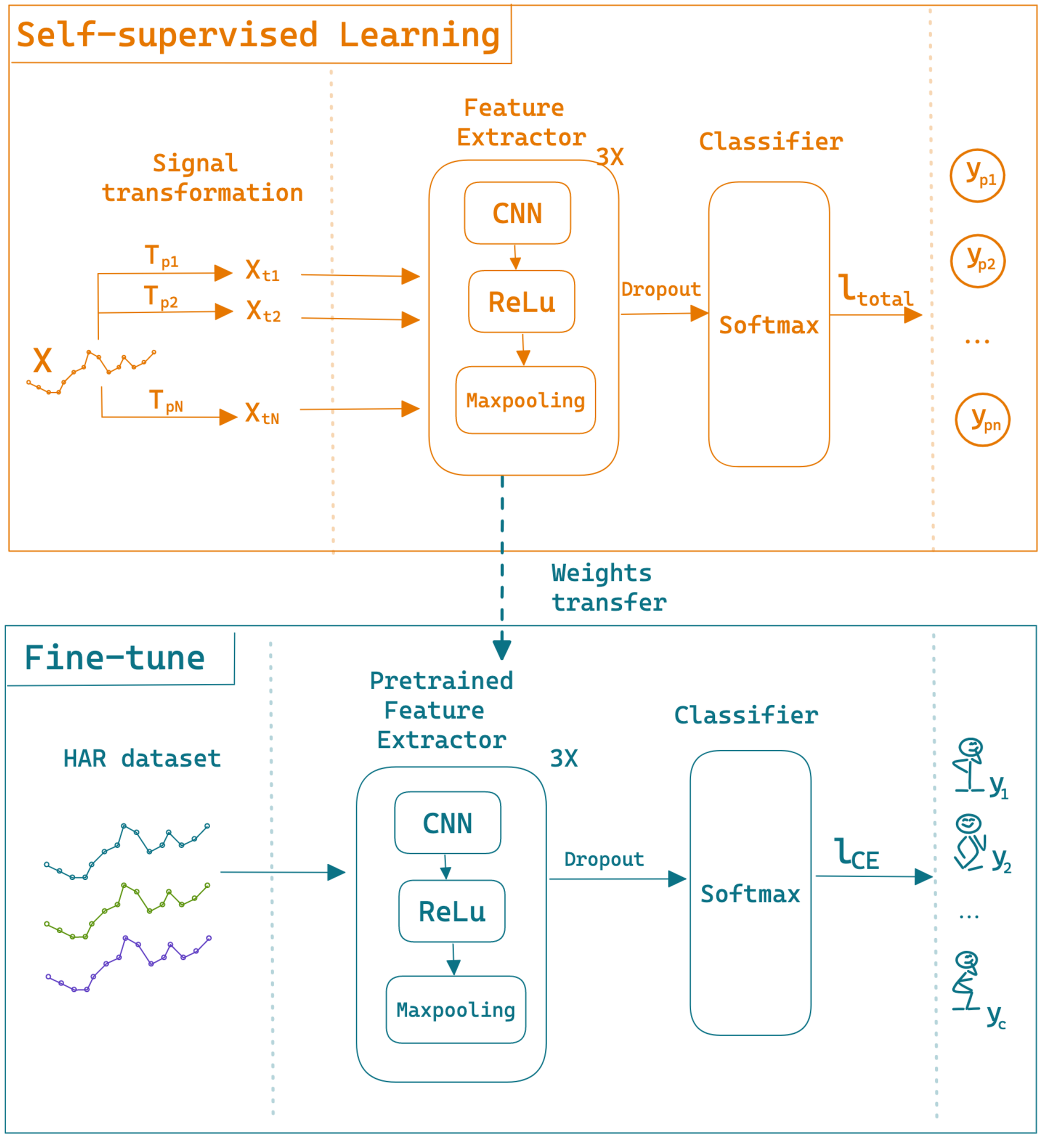

Figure 1: An overview of the proposed SSL.

celerometer sensors. The results demonstrate that smartphones and accelerometer sensors can produce more accurate classifications.

3. We addressed the label scarcity problem that tends to happen in most real-world scenarios and developed a self-supervised learning (SSL) framework based on a novel pretext task supported with a consistency loss. This framework can learn useful representations from unlabeled datasets.

4. With only **5%** of labeled samples, our pretrained self-supervised learning model was found to outperform supervised learning with full labels.

5. We find that SSL pretext task enhances the model's robustness to transformations that can occur to test data and can be robust against the data imbalance problem.

The remaining of this paper is structured as follows. Section 2 describes the previous works related to deep learning-based human activity recognition. Section 3 illustrates our proposed methodology. Section 4 describes the experimental setup and shows the results. Section 5 makes the conclusion and discusses areas for future work. Appendix A describes the datasets in detail. Appendix B lists the network hyperparameter details. Appendix C and D discuss the project resources and project schedule.

2 Related work

2.1 Machine Learning-based HAR Systems

The increased use of portable devices, such as cellular phones and watches, allowed more availability for several sensors, e.g., GPS, accelerometer, and gyroscope. These sensors provided a data-rich environment, which resulted in a surge of research studies focusing on HAR. Kwapisz et al. (2010) is one of the pioneers to utilize the WISDM Dataset with Android phone accelerometer sensors to classify 6 HAR classes: standing, walking, sitting, jogging, climbing the stairs, and walking down the stairs. The study adopted several predictive machine learning models, i.e., decision trees, logistic regression, and multi-layer perception. The experiments showed that they could classify the activities with an average accuracy of 85%. Esfahani & Malazi (2017) created the Position-Aware Multi-Sensor Dataset, which contained both accelerometer and gyroscope data. Based on different participants' bio-metric information, they applied different models to analyze their activities. The work predicted the same activities in WISDM and achieved a higher average precision of 88.5%.

2.2 Deep learning-based HAR Systems

A convolutional neural network was one of the rapidly developing Deep Learning networks (Guangle et al., 2019). Compared with its precursors, the main advantage of CNN was that it automatically detected significant features without human supervision which made it the most popular. Introduced by Hochreiter & Schmidhuber (1997), LSTM was a novel, efficient, gradient-based method, that belonged to a kind of RNN capable of learning long-term dependencies. LSTM also solved artificial long-time-lag challenges that prior RNN algorithms have never been able to accomplish. LSTM became popular because it solved the vanishing gradients problems. Preeti & Mansaf (2020) developed a hybrid lightweight RNN-LSTM algorithm, which achieved an accuracy of 95.78% on the WISDM Dataset. Additionally, Mutegeki & Han (2020) proposed an LSTM-CNN Architecture for Human Activity Recognition that achieved above 94% accuracy on the same dataset.

Bahdanau et al. (2015) proposed the first Attention model to overcome the vanishing and exploding gradient problem in RNN, which made them inefficient for longer sentences and sequences. Although LSTM can capture the longer-range dependency compared with RNN, it tended to become forgetful in specific cases. The encoder-decoder of the Attention Mechanism helped to address this issue. The objective behind using the attention mechanism was to empower the decoder to flexibly leverage the most relevant segments of the input sequence. This was achieved by computing weights for all the encoded input vectors, with the most significant vectors being assigned the highest weights. The weighted combination of these vectors then enabled the decoder to utilize the relevant information effectively.

2.3 Self-supervised Learning

In the field of self-supervised learning, several studies have been conducted to improve the performance and robustness of models trained on unlabeled data. For example, Sarkar & Etemad (2020) developed a robust SSL network to extract high-level and abstract representations from unlabelled ECG data. Additionally, Zhang et al. (2021) applied the SSL framework on time-series data and optimized the hyperparameters for better performance. Eldele et al. (2023) implemented three sleep stage classification modes to self-learn sleep datasets with few data labels and proved that SSL achieved a competitive performance compared to the supervised training that is trained with full data labels. Moreover, the pre-trained SSL also bolstered the model's robustness against issues such as data imbalance and domain shift. These studies demonstrated the effectiveness of SSL in improving the greater resilience and performance of models trained on unlabeled data in various domains, including healthcare and natural language processing(Elnaggar et al. (2021)).

While self-supervised learning has shown promising results and advantages in the ECG domain, we extended the idea to the HAR domain, which has rarely been mentioned before. We propose a self-supervised network with CNN for learning HAR features from unlabeled data. We also develop fully-supervised learning with the CNN-LSTM-Attention network and compared their result on three different human activity datasets: UCIHAR Dataset, HAPT Dataset, and HHAR Dataset. The time series data is obtained by participants via smartphones or smartwatches. In the proposed SSL approach, signal transformation tasks proposed by Um et al. (2017), is applied to input signals for learning spatiotemporal representations. Then, the weights of the pre-trained network are frozen and transferred to a class recognition network, where convolutional layers remained fixed and dense layers were trained with labeled data. We evaluate the model based on the f1-score, which takes into account both the precision score and the recall score. The experiment results 4 demonstrate that the SSL network significantly improved performance compared to a fully-supervised learning network.

3 PROPOSED METHODOLOGY

In this section, we first introduce our fully-supervised framework with full data labels, where we provide an overview of the problem and the settings, then discuss the proposed framework. Following that, we discuss the proposed self-supervised learning framework with a few data labels fine-tuning in the same hierarchy.

3.1 LIST OF SYMBOLS

We provide a list of symbols used in our discussion to ease understanding.

1. x represents the time series HAR data input.
2. y represents the true label from class C.
3. x_f represents the original features.
4. x_{tk} represents the transformed x with data augmentation.
5. x_{ftk} represents the transformed features.
6. y_a represents the the pseudo label in SSL.
7. y_{pt} represents the classification label in SSL stage 1.
8. y_p represents the classification label in fully supervised learning and SSL stage 2.

3.2 SUPERVISED LEARNING FRAMEWORK

3.2.1 OVERVIEW

Our objective is to develop a supervised learning network that can effectively learn the data representation and achieve accurate classification given the HAR datasets. The proposed network, named CNN-LSTM-Attention, comprises two consecutive 1-D CNN layers, followed by an LSTM layer, an Attention layer, a flattened layer, and a dense layer, as depicted in Figure 2 and 3. The initial CNN layer comprises a kernel size of 6, a stride of 1, and 2 paddings, followed by a ReLU activation function. Subsequently, a max-pooling function with a kernel size of 2 is applied. The input and output channels are up to different dataset settings. The second CNN layer has 64 input channels, 128 output channels, a kernel size of 3, a stride of 1, and 2 paddings. It is also followed by a ReLU activation function. Subsequently, a max-pooling function with a kernel size of 2 is applied. To prevent overfitting, we apply a 0.1 dropout rate. An LSTM network with 128 hidden sizes is employed to capture the information in both forward and backward propagation. The tanh activation function is applied after the LSTM layer. Subsequently, an attention mechanism is utilized to merge important features and select critical features by redistributing the weights and scores. Finally, a fully connected layer is used for the classification task. The model is trained for 50 epochs with a learning rate of 0.0025 and a batch size of 64. Optimizer Adam is used to optimize the learning rate of the model.

3.2.2 PROPOSED CNN-LSTM-ATTENTION NETWORK

Convolutional Neural Network

The Convolutional Neural Network (CNN) model consists of convolutional layers, activation layers, and pooling layers. Formally, let $x \in \mathbb{R}^n$ be the one-dimensional input data with shape (number of inputs) x (input length) x (input channels), where n is the length of the input sequence. The convolutional layer takes this input and applies a set of k filters, denoted by $W \in \mathbb{R}^{k\times h}$, where h is the filter size. The output of the convolutional layer can be expressed as:

$$\mathbf{y_i} = \mathbf{f}(\sum_{\mathbf{j=0}}^{\mathbf{k-1}} \mathbf{w_j x_{i+j}} + \mathbf{b}) \tag{1}$$

where y_i is the output at position i, W_j is the j-th element of the filter W, x_{i+j-1} is the j-th input element, b is a bias term, and f is an activation function such as ReLU, represented as $f(x) = max(0, x)$. ReLu is a non-linear activation function to set all negative values to zero, which helps to prevent exponential growth after the CNN computation.

After the convolutional layer, a pooling layer can be added to downsample the output, we use max-pooling to reduce the dimensionality of the features maps while preserving the important information. Now we get the sequence of feature maps Each feature map x_t is a two-dimensional matrix representing a different aspect of the input sequence.

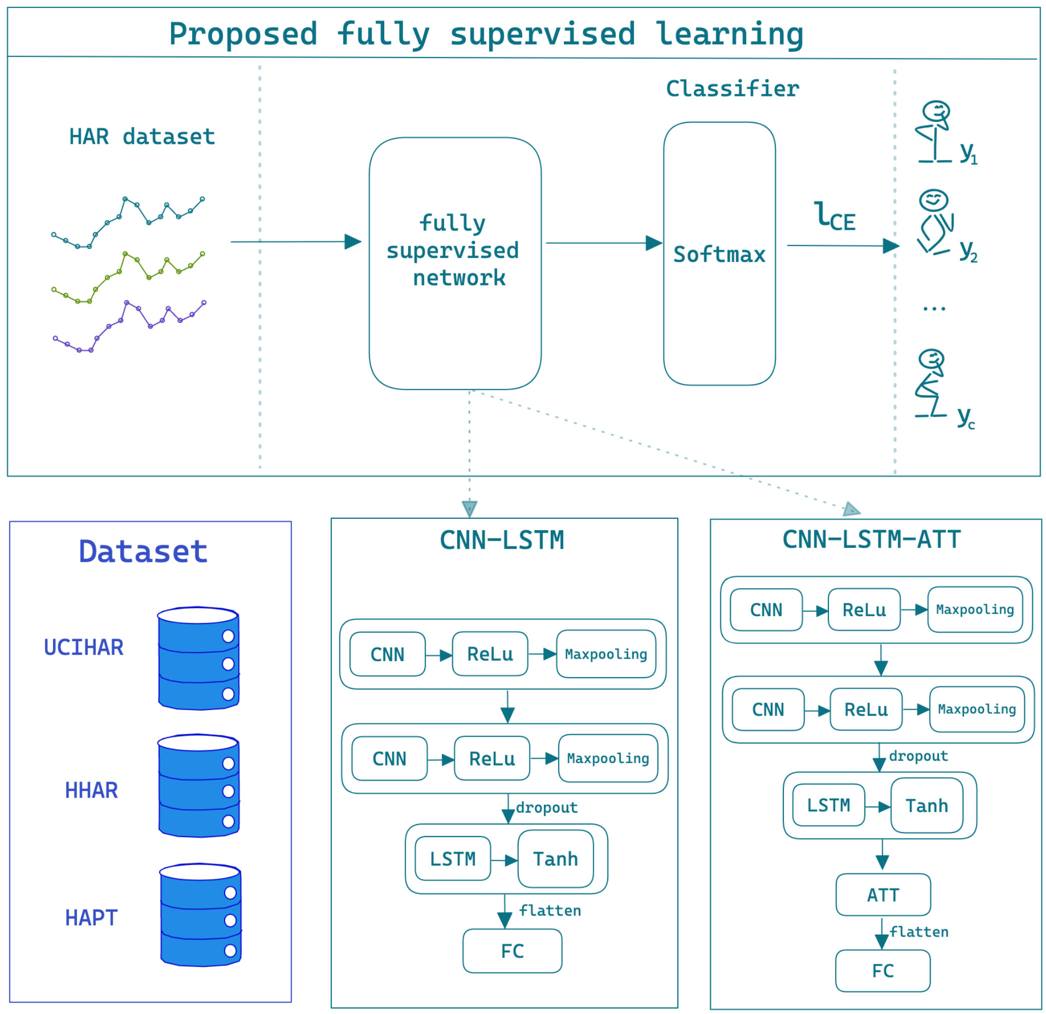

Figure 2: Architecture of CNN-LSTM-Attention network.

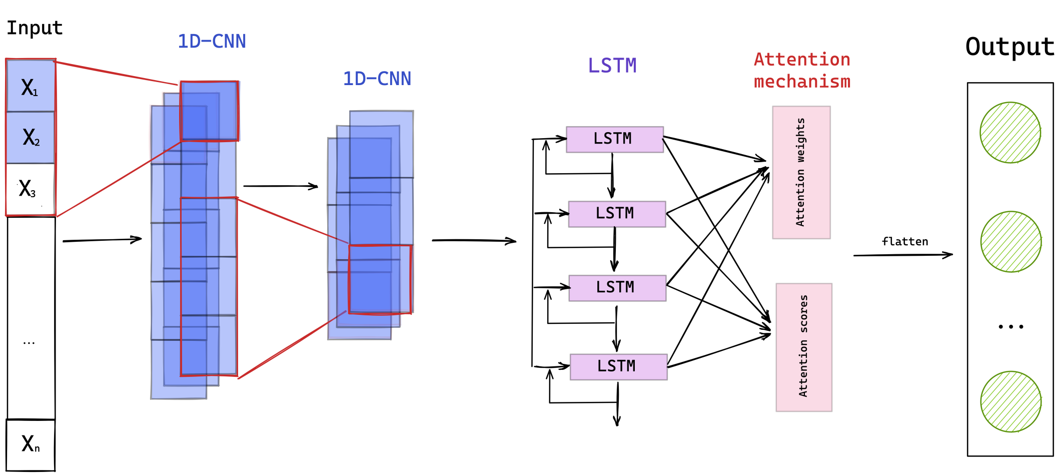

Figure 3: Detail architecture of CNN-LSTM-Attention network.

Long Shot Term Memory

The LSTM model is designed to overcome the vanishing gradient problem in traditional RNN. It uses a series of gates to control the flow of information through the cell state, which helps the model selectively remember or forget information over long sequences. The LSTM input gate, forget gate, and output gate are computed as follows (Olah, 2015):

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \tag{2}$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \tag{3}$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \tag{4}$$

where σ is the sigmoid activation function, W_{xi}, W_{hi}, b_i are the weight matrix and bias for the input gate, W_{xf}, W_{hf}, b_f are the weight matrix and bias for the forget gate, and W_{xo}, W_{ho}, b_o are the weight matrix and bias for the output gate.

Given x_t, which is passed one LSTM layer with tanh activation with 128 hidden layer sizes and generates a sequence of hidden states, the candidate cell state $\tilde{C}_t$ and the new cell state C_t are computed as:

$$\tilde{C}_t = \tanh(Wxcx_t + W_{hc}h_{t-1} + b_c) \tag{5}$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{6}$$

where $\odot$ is the element-wise multiplication operator, W_{xc}, W_{hc}, b_c are the weight matrix and bias for the candidate cell state, and C_{t-1} is the previous cell state. The tanh activation function is to determine the candidate cell state. The tanh() maps the negative inputs to strongly negative and the zero inputs to near zero. The output after the LSTM is a sequence of vectors that represent the learned features of the input sequence at each time step.

Finally, the output h_t is computed:

$$h_t = o_t \odot \tanh(C_t) \tag{7}$$

where h_t is the output at time t.

Temporal Attention Mechanism

Temporal attention mechanism(TemporalAttn) that leverages the output of an LSTM h_t to assign weights to each time step of the input sequence, with the weights being used to compute a context vector that is then fed into a fully connected layer to obtain an attention vector. TemporalAttn takes a hidden size 128 as input and initializes two fully connected layers (fc1 and fc2) with no bias. The forward method of the TemporalAttn class takes h_t as input. It first passes hidden states through fc1 and then uses h_t and the output of fc1 to calculate a score vector s_v. The score vector s_v is passed through a softmax function to obtain attention weights a_w. The attention weights a_w are then used to calculate a context vector c_v as a weighted sum of the h_t. This context vector c_v is concatenated with the h_t and passed through fc2 to obtain an attention vector a_v. The attention vector a_v is then passed through a tanh function and returned along with the attention weights a_w.

Last but not least, the resulting attention vector a_v is passed to the dense layer to classify 6 or 12 classes y_p depending on the dataset. In the supervised learning approach, the model is trained and tested with full data labels.

The details hyperparameters are listed in Appendix Table 13.

Loss function - Cross-Entropy Loss

Criterion loss is typically used in classification tasks. We use the cross-entropy loss as the following formula:

$$\mathcal{L}_{\text{CrossEntropy}} = ((1 - y_p)\log(1 - p)) - (y_p \log(p)) \tag{8}$$

Where y_p denotes the binary indicator (0 or 1) that whether the classification y_p belongs to the y class, and p denotes the classification probability that y_p belongs to y class.

3.3 Self-supervised Learning

3.3.1 Overview

Our aim is to learn the HAR features and representation among the classes in an unsupervised method. We proposed the self-supervised network which consists of two stages: 1) self-supervised learning of activity; 2) learning to classify activity. Criterion loss is used in classification tasks as the following formula:

$$\mathcal{L}_{\text{CrossEntropy}} = ((1 - y_{pt,j})\log(1-p)) - (y_{pt,j}\log(p)) \tag{9}$$

Where $y_{pt,j}$ denotes the binary indicator (0 or 1) that whether the SSL classification y_{pt} belongs to the $y_a j_{th}$ class, and p denotes the classification probability that $y_{pt,j}$ belongs to $y_a j_{th}$ class.

Formally, we design two types of tasks, pretext task T_p and downstream task T_d. T_p represents the pre-trained tasks to learn the data representation without access to the data label$y$. With Um et al. (2017)'s work on different data augmentation techniques, we introduce four techniques in this paper: permeation, time shift, scaling, and add noise. The effectiveness of these augmentation techniques has also been proved in Saeed et al. (2019), and Eldele et al. (2023). While T_p represents the activity classification, the model has access to the input x and data label y in T_p. In this paper, classifying the activity classes is the downstream task, such as classes walking, walking upstairs, walking downstairs, sitting, standing, and laying in UCIHAR. Figure 4 shows the proposed self-supervised learning network. In the following subsections, we discuss the detailed implementation.

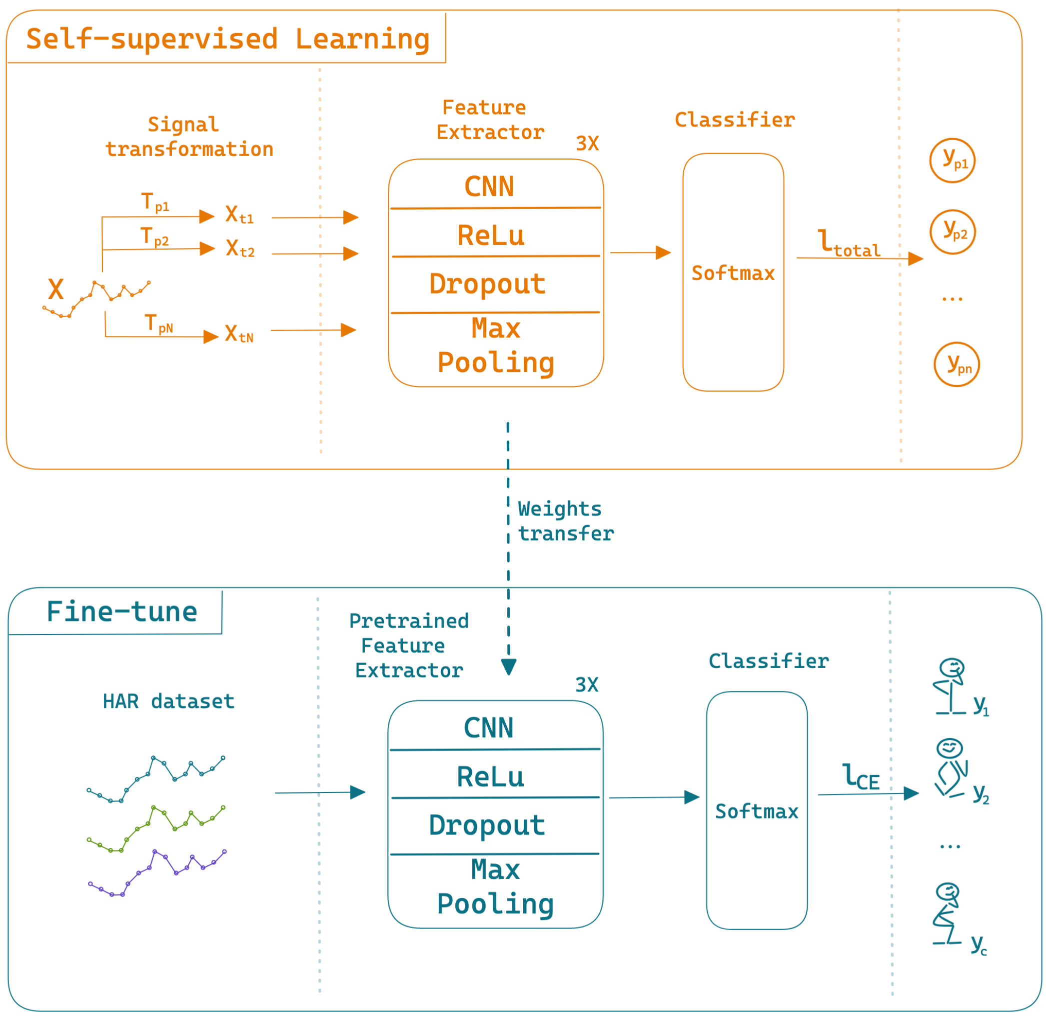

Figure 4: Architecture of self-supervised learning network.

3.3.2 Stage 1: Pretext Task

In this stage, the model is trained on a full unlabeled dataset to learn the data representations. In data reprocessing, T_p is performed. x is transformed to k^{th} transformed signal x_{tk} with self-generate pseudo label y_a. $k \in [0, N]$ while N is the number of signal transformations.

Signal transformation

We applied four data augmentation techniques on the x to generate new x_{tk} by k^{th} transformations.

Permutation: it randomly changed the temporal position. To do this, we divided the data into five segments that have the same length, and then we shuffled them in a random order to make a x_{tk}.

Time shift: similar to permutation, it squished or stretches on the randomly selected segments x to x_{tk}.

Scale: it changed the data size by scaling x in random scalar to x_{tk}.

Add noise: it added random values as noise to x. Here we normalized x to x_n, then sum x_n and x to x_{tk}.

In self-supervised learning, data augmentation can be used to create diverse and realistic variations of the input data without explicit labels. The model learns to recognize the same object or pattern under different transformations, making the model robust and reducing overfitting.

Formally, we perform self-supervised learning of activity: $\delta(x, x_{tk}, y_a, \lambda)$. The model minimizes the total loss by learning the λ as training parameters, where δ represents the self-supervised learning of activity.

There are feature extractor and classifier in $\delta(x, x_{tk}, y_a, \lambda)$. The architecture of the 1D-CNN network comprises three convolutional blocks. Each block consists of a Convolutional layer followed by a non-linear activation function ReLU, and a MaxPooling layer. It drops out by 0.1 after the third CNN layer. The hyperparameters setup is listed in Appendix Table 14.

We use the 1D-CNN as the feature extractor. First, pass (x) to the feature extractor. After the dense layer, we get the original features x_f: $(x) \rightarrow x_f$; pass x_{tk} to the feature extractor. After the dense layer, we get the transformed features x_{ftk}: $x_{tk} \rightarrow x_{ftk}$. After stage 1 training, the weights of SSL will be passed to stage 2 denotes as λ.

Consistency loss - MSE loss

Motivated by Xie et al. (2020)'s work, we aim to calculate consistency loss in self-supervised learning which enhances the robustness of the learned features. In self-supervised learning, the model is trained to predict certain properties of the data without human annotations. However, due to the unsupervised nature of the training, the model may learn features that are not robust and do not generalize well to unseen data. Consistency loss is a regularization technique that encourages the model to produce consistent predictions when the input is perturbed in some way.

With the experiment 4.2.2, we apply MSE loss to measure the average squared difference between the original features x_{tk} and transformed features x_{ftk}. It is commonly used in regression tasks and can be computed using the following formula:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{n}\sum_{i=1}^{n}(x_{tk} - x_{ftk})^2 \tag{10}$$

Where n is the number of samples. Next, in classification, x_{ftk} is classified to predict the label y_{pt}: $x_{ftk} \rightarrow y_{pt}$. We can compute the Cross-Entropy loss of y_{pt} and y_a, and sum with the MSE loss of x_{tk} and x_{ftk}. The result contributes to consistency loss per training.

3.3.3 Stage 2: Downstream Task - Learning to Classify Activity

In this stage, the model is fine-tuning with the data-scarce scenario with true label y. We have the feature extractor and classifier as well: $\varepsilon(x, y, \lambda)$. While the λ is the frozen weight that is transferred from stage 1. The cross-entropy loss is computed for classification y_p and y.

We use 1D-CNN with λ weight as the feature extractor in this stage. First, we pass (x) to the feature extractor. After the dense layer, we get the original features x_f: $x \rightarrow x_f$. Next, in classification, x_f is classified to predict the label $y_p : x_f \rightarrow y_p$. The cross-entropy loss of y_p and true label y is computed. In the SSL approach, we focus on classification in scenarios where the amount of available data is limited. Therefore, the stage 2 model is fine-tuned with few label data percentages to simulate real-world situations.

4 Experimental Results

We evaluate the model on three publicly available datasets, namely the UCIHAR dataset, HAPT dataset, and HHAR dataset. The performance of these models is evaluated by f1-score and test accuracy. We maintain a 70% training and 30% testing ratio on the experiments. While 15% of training data is taken on validation. First, we conduct tests on the supervised CNN-LSTM model, the supervised CNN-LSTM-Attention model, and 1-D CNN self-supervised learning model across the datasets with baseline comparison. Second, we conduct the experiment with data imbalanced issues on HAPT datasets. Last, we conduct an experiment on the HHAR phone and watch dataset for the same user to evaluate which type of device and sensor performs better.

4.1 Dataset summary

UCIHAR

The UCIHAR Machine Learning Repository collected the data from the smartphones(Samsung Galaxy S II) embedded accelerometer and gyroscope sensors for six activities: walking, walking upstairs, walking downstairs, sitting, standing, and laying (Jorge et al., 2012). Each subject was collected from thirty participants within the twenty-two to seventy-nine years age group to perform six activities for sixty seconds while wearing a smartphone around the waist. The sensor data was collected at the 3-axial linear acceleration and 3-axial angular velocity at a constant rate of 50Hz. The sensor signals were well pre-processed by noise filters and sampled in fixed 2.56 seconds sliding windows and 50% overlapped with 128 readings window.

HAPT

The UCI Machine Learning Repository extended the data collection of the UCIHAR dataset. They collected six more postural transition activities in this experiment (Jorge et al., 2015). A total of twelve activities were recorded: walking, walking upstairs, walking downstairs, sitting, standing, laying, stand-to-sit, sit-to-stand, sit-to-lie, lie-to-sit, stand-to-lie, and lie-to-stand. Each subject was collected from thirty participants within the nineteen to forty-eight years age group to perform six activities for sixty seconds while wearing a smartphone around the waist. The sensor data was collected at the 3-axial linear acceleration and 3-axial angular velocity at a constant rate of 50 Hz. The sensor signals were pre-processed by noise filters and sampled in fixed 2.56 seconds sliding windows and 50% overlapped with 128 readings window.

HHAR

The HHAR Machine Learning Repository collected the data from the smartphones and smartwatches embedded with accelerometer and gyroscope sensors for six activities: biking, walking, stair up, stair down, standing, and sitting(Allan et al., 2015). Each subject was collected from nine participants, Data was collected from thirty-one smartphones, four smartwatches, and one tablet by four manufacturers, running variants of Android and IOS.

More details can be found in the appendix A.

4.2 Proposed Methods Results

4.2.1 Supervised Model Results

We aim to evaluate the effect of the Attention Mechanism. We evaluate the f1-score and test Accuracy with CNN-LSTM and CNN-LSTM-Attention network on three datasets in Table 1. The better result is bold. From figure 10 Table 11 and 12 we can imply that the classes in UCIHAR and HHAR are well-balanced, while there is an imbalanced issue in the HAPT dataset, which we will discuss in 4.3. In this experiment, the HAPT dataset is upsampling with a replication technique.

	CNN-LSTM		CNN-LSTM-Attention	
Mean	F1-Score(%)	Accuracy(%)	F1-Score(%)	Accuracy(%)
UCIHAR	**91.56%**	**91.48%**	88.10%	88.29%
HAPT_replication	**83.81%**	**92.81%**	80.37%	89.49%
HHAR	**92.05%**	**91.62%**	66.63%	68.61%

Table 1: Comparison of models on three Datasets using smartphone data, specifically accelerometer and gyroscope data. Replication is applied to the HAPT dataset.

Mean F1-Score(%)	Mutegeki & Han (2020)	Sa-nguannarm et al. (2021)	Ours work
UCIHAR	91.55%	-	**91.56%**
HHAR	-	86.50%	**92.05%**

Table 2: Proposed supervised CNN-LSTM network comparison with baselines.

Discussion

We can conclude that the CNN-LSTM model performs better than the CNN-LSTM-Attention model with the stated datasets, in which the f1-score is higher than another. From the experiment result, the attention mechanism did not improve the performance of the CNN-LSTM model on time series data. The reason is the CNN-LSTM model already captured the most relevant features with one-dimensional data, making the addition of an attention mechanism unnecessary. Another reason is the insufficient diversity of data. Attention mechanisms are often used in natural language processing (NLP) and computer vision tasks, where the input data is multi-dimensional, as mentioned in Vaswani et al. (2017)'s work. The time series data already has a natural order and structure that can be captured by the LSTM layer in the model. Thus, the need for an attention mechanism may not be as significant. Hence, based on the characteristics of the data and the task at hand discussed in this paper, we suggest using the CNN-LSTM network in a supervised approach.

Compared with the baseline in Table 2, our CNN-LSTM model achieved comparable performance in the UCI-HAR dataset and outperformed 5.55% in the HHAR dataset. Figure 5 plots the train and test accuracy trend on three datasets.

4.2.2 Self-supervised Learning Results

We develop a 1D-CNN SSL network. We pre-train the network with four transformations: permutation, time shift, scaling, and adding noise. After that, we fine-tune the network with varying degrees of data scarcity, randomly selecting 5%,10%, 20%, and 50% of the full dataset. We compare the fine-tuning SSL performance with limited data and the fully-supervised learning network with full labels. Additionally, we analyze how the different loss functions can enhance classification performance.

In the first stage: self-supervised learning of activity with 100% of the unlabeled training data sample is passed to the model, and pseudo labels are generated. In the second stage: learning to classify activity, we fine-tune

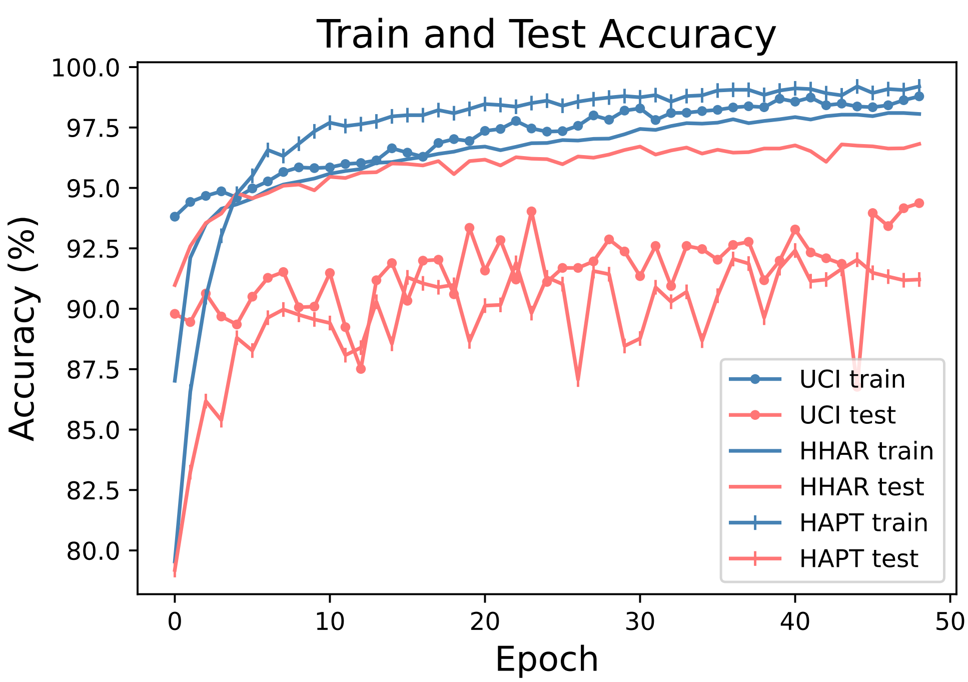

Figure 5: CNN-LSTM network train and test accuracy on three datasets.

the model with 5%, 10%, 20%, and 50% data samples and labels. This is to simulate real-world data-hungry scenarios. Where HAPT_raw represents the HAPT dataset with imbalanced classes. The SSL model fine-tuning result is shown in Table 3. We evaluated the comparison with the best f1-score% and best test accuracy%.

Dataset	Best	5% FT.	10% FT.	50% FT.	100% Sup.(CNN-LSTM)
UCIHAR	F1-Score(%)	**91.73%**	93.02%	96.36%	91.5%
	Accuracy(%)	**91.69%**	92.83%	96.10%	91.13%
HAPT_raw	F1-Score(%)	**85.16%**	86.38%	89.85%	84.42%
	Accuracy(%)	**86.46%**	89.65%	89.73%	85.45%
HHAR	F1-Score(%)	67.16%	**96.38%**	89.85%	92.4%
	Accuracy(%)	87.18%	**93.30%**	96.48%	92.79%

Table 3: Comparison of the SSL performance on three datasets. Sup. stands for supervised CNN-LSTM training performance, while FT. refers to fine-tuning performance.

UCIHAR Mean	F1-Score(%)	Accuracy(%)
SSL(Saeed et al. (2019))	88.90%	87.50 %
TS-TCC(Eldele et al. (2021))	**90.38%**	**90.37%**
SSL-ECG(Sarkar & Etemad (2020))	63.73%	65.34%
Ours work	89.83%	89.87%

Table 4: Baselines and comparison on UCIHAR dataset.

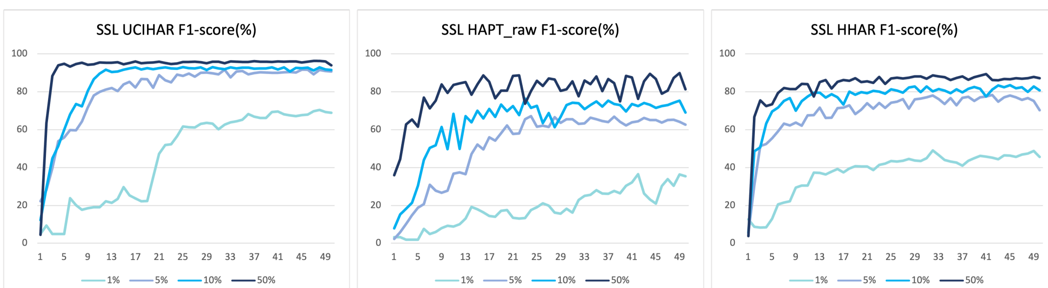

Figure 6: Fine-tuning the pretrained SSL networks on three dataset - f1-score(%).

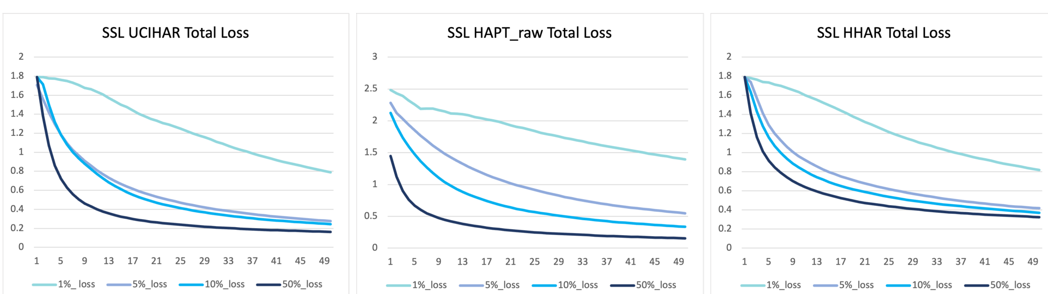

Figure 7: Fine-tuning the pretrained SSL networks on three dataset - total loss.

Discussion

Figure 6 and 7 outline the SSL f1-score and total loss trends on three datasets. In Table 3, our study finds that fine-tuning only **5%** label data, 1-D CNN self-supervised learning network performs better than CNN-LSTM fully-supervised learning with full-label data training. This finding is solid in UCIHAR and HAPT datasets. While on HAPT imbalanced datasets, SSL fine-tuning with **10%** labeled data result is higher than supervised learning by 3.92%. The best performance score is bold. This result strongly proves the ability of SSL to learn the data representative itself instead of labels, which is beneficial in the real-world data-scarce scenario.

Additionally, we compare our SSL result with the previous SSL network(Saeed et al. (2019)), TS-TCC(Eldele et al. (2021)), and SSL-ECG(Sarkar & Etemad (2020)) on the UCIHAR dataset, result in Table 4. Our work achieves the second-best performance with the same 1-CNN network, our work outperforms 0.93% with the SSL algorithm, 26.1% with the SSL-ECG algorithm, and only 0.55% away from the best work TS-TCC algorithm.

Loss function

We utilize the loss function to enhance network consistency by aggregating feature loss and incorporating it with classification cross-entropy loss on the **5%** HAPT dataset. Specifically, we conduct experiments with both MSE and KLD functions and compare the results with the standard cross-entropy loss function.

HHAR	FT.	1%	5%	10%	50%
Criterion	F1-Score(%)	45.84%	69.72%	75.63%	85.51%
MSE + Criterion	F1-Score(%)	**46.05%**	**70.08%**	**76.77%**	**85.72%**
KLD + Criterion	Accuracy(%)	51.11%	71.10%	76.90%	85.87%

Table 5: Comparison of the SSL performance with MSE loss and KLD loss.

Discussion

We investigate the effect of summing MSE loss or KLD loss to the criterion loss on SSL classification performance. The Table 5 results show that summing up MSE loss and criterion loss can improve the classification efficiency, but adding KLD loss did not have a positive impact. This is because of too many zeros in the input tensor, which did not contribute to the loss calculation. In order to enhance the robustness of the network, we opted to utilize the MSE loss function on the features. We applied the consistency loss to all the SSL experiments.

4.3 HAPT Data Imbalance Problem

Our analysis of the dataset reveals an issue with class imbalance. Some experiments ignored this challenge, resulting in poor performance for minority classes, even though the overall test accuracy remains high. For instance, Zhang et al. (2018) fed the model with raw HAPT samples despite the data imbalanced issue; Jain et al. (2022) only measured the classes with sufficient sample numbers but abandoned the imbalanced classes; Thu & Han (2021) grouped the insufficient sample numbers classes as two other groups: "PT1" and "PT2".

However, in some cases, the performance of minority classes is the critical metric. Figure 8 illustrates the distribution of each activity before and after applying the oversampling technique, highlighting the scarcity of data for these classes. Figure 9 presents the confusion matrix for the imbalanced data, which reveals a score of zero for classes with only a few samples, such as stand-to-sit, sit-to-stand, sit-to-lie, lie-to-sit, stand-to-lie, and lie-to-stand. We address this issue by using oversampling techniques, which increase the number of samples in each activity to the original maximum count.

In this paper, we use two techniques to upsample the minority classes to address the imbalanced dataset problem: Synthetic Minority Oversampling Technique(SMOTE) and oversampling by replication. SMOTE functions by selecting nearby examples in the feature space, drawing a line linking the examples in the feature space, and then drawing a new sample along the line(Chawla et al., 2002). We apply SMOTE technique to the raw data and the saved balanced data file will be loaded in the training process. In another way, oversampling by replication technique is applied to data, which finds the class with the maximum count, and randomly selects an upsample in each class equal to that maximum number. This method is called when run-time after the training data is loaded. Both of them involve synthesizing the existing minority class examples, although these examples don't add any new information to the model. We evaluate the mean f1-score and test accuracy of the balanced HAPT dataset with both the CNN-LSTM and CNN-LSTM-Attention models, as presented in Table 6.

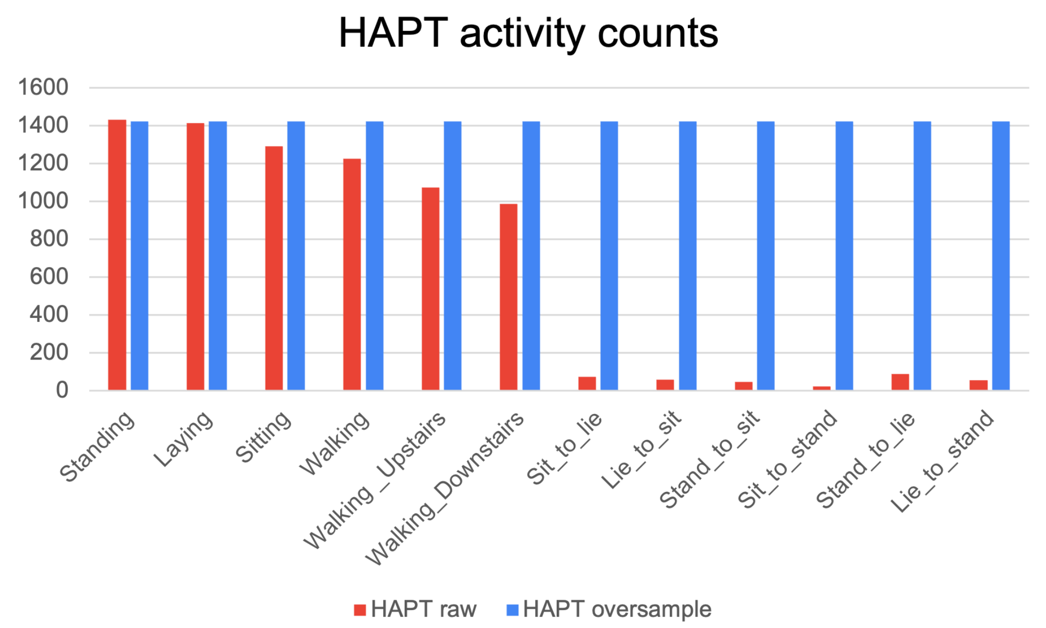

Figure 8: HAPT activity counts before and after replication.

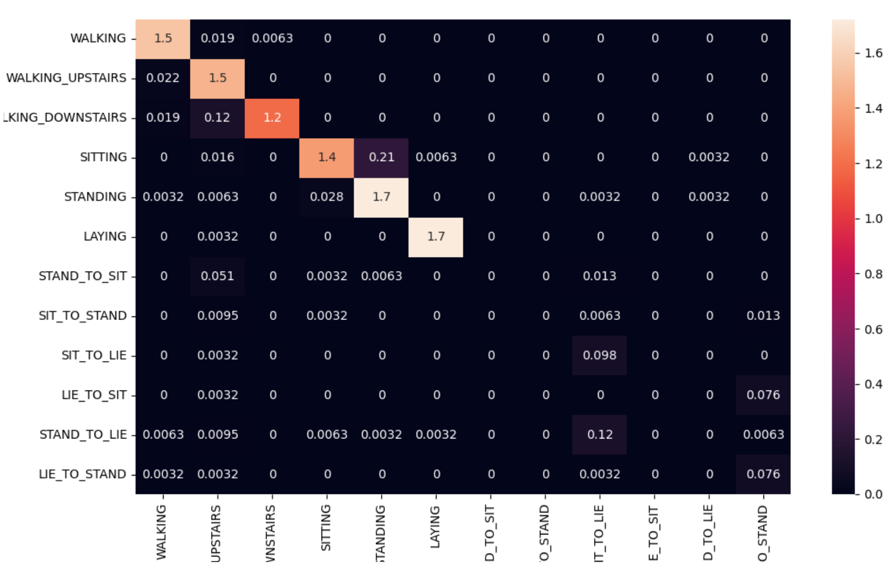

Figure 9: HAPT(imbalanced) confusion matrix.

Models	F1-Score(%)	Accuracy(%)
CNN-LSTM (SMOTE)	**84.84%**	**93.38%**
CNN-LSTM-Attention (SMOTE)	78.98%	88.99%
CNN-LSTM (replication)	**83.81%**	92.81%
CNN-LSTM-Attention (replication)	80.37%	89.49%

Table 6: Comparison of our proposed models with different methods to address the data imbalance problem on HAPT dataset.

Discussion

Regarding the data imbalanced processing, we recommend using oversampling by replication method to address the data imbalanced issue. The f1-score and test accuracy are computed on the balanced dataset that applied SMOTE and oversampling by replication method, where we find that the results are quite close between the two techniques. SMOTE is applied in the data preparation before data training while oversampling by replication method is applied when data training. We notice that SMOTE is more computationally expensive.

HAPT	Method	Stand_to_sit	Sit_to_stand	Sit_to_lie	Lie_to_sit	Stand_to_lie	Lie_to_stand	F1-Score(%)	Accuracy(%)
Raw	5%FT.	**82.35%**	**80.00%**	**47.62%**	**31.58%**	**37.50%**	22.22%	**70.55%**	**88.01%**
Oversample	5%FT.	69.57%	75.00%	40.00%	13.33%	43.75%	**24.00%**	70.01%	72.12%
Raw	100%Sup.	0	0	46.16%	0	0	0	16.54%	51.74%

Table 7: Comparison of the SSL performance on HAPT dataset without oversampling.

SSL effect - Robustness

The HAPT dataset has a very limited number of data samples for these classes, including stand-to-sit, sit-to-stand, sit-to-lie, lie-to-sit, stand-to-lie, and lie-to-stand. We investigate the effectiveness of self-supervised learning in dealing with imbalanced class data in the **5%** HAPT dataset. While the performance for the balanced classes is extremely good as expected, in this experiment, we focus on the imbalanced classes' results. The f1-score results are presented in Table 7. We compare the result of SSL fine-tuning with 5% label data on HAPT raw data, 5% label data on HAPT raw data on HAPT oversampled data, and supervised learning on HAPT raw data. The higher f1-score is bold that the SSL with the raw dataset achieves higher performance scores, while 0 scores in a few sample classes. This is because the oversampling technique is artificially replicating the sample numbers without adding new data features or diversity. While SSL learns the feature representatives themselves, rather than the sample size. It highlights that SSL is capable of handling imbalanced data robustly, which is consistent with previous Liu et al. (2021) and Eldele et al. (2023)'s work.

4.4 WHICH SENSOR IS MORE EFFICIENT?

We utilize the HHAR benchmark dataset consisting of accelerometer and gyroscope data acquired from smartwatches or smartphones. We conduct experiments using the public HHAR dataset that had been meticulously manipulated. Experiments are performed on the dataset to distinguish 6 distinct human activities with the supervised CNN-LSTM model and supervised CNN-LSTM-Attention model. From Table 11 and Table 12 we identify that the classes are very well balanced.

HHAR	CNN-LSTM		CNN-LSTM-ATT	
	F1-Score(%)	Accuracy(%)	F1-Score(%)	Accuracy(%)
smartphone-accelerometer	**95.30%**	95.49%	66.63%	68.92%
smartphone-gyroscope	88.79%	87.74%	66.63%	68.92%
smartwatch-accelerometer	**80.66%**	82.17%	67.49%	68.92%
smartwatch-gyroscope	71.72%	72.98%	67.50%	68.92%

Table 8: Comparison of HHAR dataset with different devices and different sensors.

Discussion

Table 8 compares the model with different devices and sensors. We observe that accelerometer-based classifications achieve higher accuracy than those based on gyroscopes. While in Table 9 we drill down on the user level to compare the devices and sensor's performance. The variance reveals that the phone classification result

User	Phone	Watch accelorometer	Variance	Phone	Watch gyroscope	Variance
0	83.90%	84.11%	-0.21%	84.11%	84.19%	-0.08%
1	92.90%	82.56%	10.34%	82.56%	79.98%	2.58%
2	91.81%	88.75%	3.06%	88.75%	86.91%	1.84%
3	95.91%	87.37%	8.54%	87.37%	83.93%	3.44%
4	93.86%	74.27%	19.59%	74.27%	80.73%	-6.46%
5	97.38%	88.77%	8.61%	88.77%	85.24%	3.53%
6	78.64%	88.03%	-9.39%	88.03%	82.04%	5.99%
7	97.04%	91.09%	5.95%	91.09%	86.83%	4.26%
8	92.39%	89.99%	2.40%	89.99%	88.04%	1.95%

Table 9: Comparison on different users of models on HHAR Dataset from smartwatch-gyroscope data. Variance equals the difference between Phone and Watch.

is better than the watch's results. Based on these findings, we conclude that smartphones with accelerometers are better suited for HAR classification.

5 Conclusion

In this paper, we developed a hybrid CNN-LSTM network for supervised learning that can achieve high performance in HAR classification. In order to overcome the challenges of limited data and data imbalance issues in real-world scenarios, we explored the use of simple 1D-CNN self-supervised learning and proved SSL efficacy. Comparing the previous work, our study demonstrated the effectiveness in addressing these challenges and achieving promising results in HAR classification. Moreover, we utilized consistency loss measures and multi-task signal transformation to enhance the robustness of the SSL model. In our study, we proved that SSL is beneficial to real-world data shortage situations. In future research, it is anticipated that more advanced deep learning architectures, such as Multi-Layer Perceptrons, will be employed to perform self-supervised learning. Furthermore, we aim to extend SSL learning to other types of data beyond text or images, such as graphs, audio, and videos. Moreover, we hope to extend the current knowledge to other domains to tackle the domain shift problem in transfer learning.

6 Acknowledgement

I would like to express my deepest gratitude to Assoc Prof. Kwoh Chee Keong for his patience, mentorship, and unwavering support. His expertise and dedication have been a constant source of inspiration and motivation for me.

I would also like to extend my sincere thanks to SCSE Ph.D. student Emadeldeen Eldele for his invaluable guidance and support throughout this research project. He always gives insightful direction, sufficient feedback, and encouragement to me in the past year.

Finally, my appreciation goes to the NTU End-Note sponsor, whose generous support provided us with unlimited access to reference papers crucial to this work.

REFERENCES

Stisen Allan, Blunck Henrik, Bhattacharya Sourav, Prentow Thor, Siiger, Kjærgaard Mikkel, Baun, Dey Anind, Sonne Tobias, and Mads Møller Jensen. Heterogeneity activity recognition data set, 2015.

Umran Alrazzak and Bassem Alhalabi. A survey on human activity recognition using accelerometer sensor. In *2019 Joint 8th International Conference on Informatics, Electronics & Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*, pp. 152–159, 2019. doi: 10.1109/ICIEV.2019.8858578.

Laith Alzubaidi, Jinglan Zhang, Amjad J Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, José Santamaría, Mohammed A Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions. *Journal of big Data*, 8:1–74, 2021.

Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge L Reyes-Ortiz. Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In *Ambient Assisted Living and Home Care: 4th International Workshop, IWAAL 2012, Vitoria-Gasteiz, Spain, December 3-5, 2012. Proceedings 4*, pp. 216–223. Springer, 2012.

Ferhat Attal, Samer Mohammed, Mariam Dedabrishvili, Faicel Chamroukhi, Latifa Oukhellou, and Yacine Amirat. Physical human activity recognition using wearable sensors. *Sensors*, 15(12):31314–31338, 2015.

Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations, ICLR*, January 2015. 3rd International Conference on Learning Representations, ICLR 2015 ; Conference date: 07-05-2015 Through 09-05-2015.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

John Edwards. Wireless sensors relay medical insight to patients and caregivers [special reports]. *IEEE Signal Processing Magazine*, 29(3):8–12, 2012. ISSN 1053-5888. doi: 10.1109/msp.2012.2183489. URL `https://ieeexplore.ieee.org/document/6179818/`.

Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee Keong Kwoh, Xiaoli Li, and Cuntai Guan. Time-series representation learning via temporal and contextual contrasting. *arXiv preprint arXiv:2106.14112*, 2021.

Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee-Keong Kwoh, and Xiaoli Li. Self-supervised learning for label-efficient sleep stage classification: A comprehensive evaluation. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 2023.

Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rehawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, et al. Prottrans: Toward understanding the language of life through self-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 44 (10):7112–7127, 2021.

Pegah Esfahani and Hadi Tabatabaee Malazi. Pams: A new position-aware multi-sensor dataset for human activity recognition using smartphones. In *2017 19th International Symposium on Computer Architecture and Digital Systems (CADS)*, pp. 1–7, 2017. doi: 10.1109/CADS.2017.8310680.

Munkhjargal Gochoo, Tan-Hsu Tan, Shing-Hong Liu, Fu-Rong Jean, Fady S Alnajjar, and Shih-Chia Huang. Unobtrusive activity recognition of elderly people living alone using anonymous binary sensors and dcnn. *IEEE journal of biomedical and health informatics*, 23(2):693–702, 2018.

Yao Guangle, Lei Tao, and Zhong Jiandan. A review of convolutional-neural-network-based action recognition. *Pattern Recognition Letters*, 118:14–22, 2019. ISSN 0167-8655. doi: https://doi.org/10.1016/j.patrec.2018.05.018. URL `https://www.sciencedirect.com/science/article/pii/S0167865518302058`. Cooperative and Social Robots: Understanding Human Activities and Intentions.

Saurabh Gupta. Deep learning based human activity recognition (har) using wearable sensor data. *International Journal of Information Management Data Insights*, 1(2):100046, 2021.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, nov 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL `https://doi.org/10.1162/neco.1997.9.8.1735`.

Rahul Jain, Vijay Bhaskar Semwal, and Praveen Kaushik. Deep ensemble learning approach for lower extremity activities recognition using wearable sensors. *Expert Systems*, 39(6):e12743, 2022.

L. Reyes-Ortiz Jorge, Anguita Davide, Ghio Alessandro, Oneto Luca, and Parra Xavier. Human activity recognition using smartphones data set. https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones, 2012.

L. Reyes-Ortiz Jorge, Anguita Davide, Ghio Alessandro, Oneto Luca, and Parra Xavier. Smartphone-based recognition of human activities and postural transitions data set. https://archive.ics.uci.edu/ml/datasets/smartphone-based+recognition+of+human+activities+and+postural+transitions, 2015.

Bojan Kolosnjaji and Claudia Eckert. Neural network-based user-independent physical activity recognition for mobile devices. In *Intelligent Data Engineering and Automated Learning–IDEAL 2015: 16th International Conference, Wroclaw, Poland, October 14-16, 2015, Proceedings 16*, pp. 378–386. Springer, 2015.

Jennifer R. Kwapisz, Gary M. Weiss, and Samuel A. Moore. Activity recognition using cell phone accelerometers. In *Proceedings of the Fourth International Workshop on Knowledge Discovery from Sensor Data*, pp. 10–18, 2010.

Xinyu Li, Yuan He, and Xiaojun Jing. A survey of deep learning-based human activity recognition in radar. *Remote Sensing*, 11(9):1068, 2019.

Hong Liu, Jeff Z HaoChen, Adrien Gaidon, and Tengyu Ma. Self-supervised learning is more robust to dataset imbalance. *arXiv preprint arXiv:2110.05025*, 2021.

Jeffrey W. Lockhart, Gary M. Weiss, Jack C. Xue, Shaun T. Gallagher, Andrew B. Grosner, and Tony T. Pulickal. Design considerations for the wisdm smart phone-based sensor mining architecture, 2011. URL https://doi.org/10.1145/2003653.2003656.

Ronald Mutegeki and Dong Seog Han. A cnn-lstm approach to human activity recognition. In *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pp. 362–366, 2020. doi: 10.1109/ICAIIC48513.2020.9065078.

Christopher Olah. Understanding lstm networks, Aug 2015. URL https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Sandeep Kumar Polu and SK Polu. Human activity recognition on smartphones using machine learning algorithms. *International Journal for Innovative Research in Science & Technology*, 5(6):31–37, 2018.

Agarwal Preeti and Alam Mansaf. A lightweight deep learning model for human activity recognition on edge devices. In *International Conference on Computational Intelligence and Data Science*, volume 167, pp. 2364–2373, 2020. doi: https://doi.org/10.1016/j.procs.2020.03.289. URL https://www.sciencedirect.com/science/article/pii/S1877050920307559.

Charissa Ann Ronao and Sung-Bae Cho. Human activity recognition using smartphone sensors with two-stage continuous hidden markov models. In *2014 10th international conference on natural computation (ICNC)*, pp. 681–686. IEEE, 2014.

Phataratah Sa-nguannarm, Ermal Elbasani, Bongjae Kim, Eung-Hee Kim, and Jeong-Dong Kim. Experimentation of human activity recognition by using accelerometer data based on lstm. In *Advanced Multimedia and Ubiquitous Engineering: MUE-FutureTech 2020*, pp. 83–89. Springer, 2021.

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.

Pritam Sarkar and Ali Etemad. Self-supervised ecg representation learning for emotion recognition. *IEEE Transactions on Affective Computing*, 13(3):1541–1554, 2020.

Satya P Singh, Madan Kumar Sharma, Aimé Lay-Ekuakille, Deepak Gangwar, and Sukrit Gupta. Deep convlstm with self-attention for human activity decoding using wearable sensors. *IEEE Sensors Journal*, 21(6): 8575–8582, 2020.

Akara Supratak. deepsleepnet. https://github.com/akaraspt/deepsleepnet/, 2020.

Nguyen Thi Hoai Thu and Dong Seog Han. Hihar: A hierarchical hybrid deep learning architecture for wearable sensor-based human activity recognition. *IEEE Access*, 9:145271–145281, 2021.

Terry T Um, Franz MJ Pfister, Daniel Pichler, Satoshi Endo, Muriel Lang, Sandra Hirche, Urban Fietzek, and Dana Kulić. Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks. In *Proceedings of the 19th ACM international conference on multimodal interaction*, pp. 216–220, 2017.

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.

Jindong Wang. Deep-learning-activity-recognition. `https://github.com/jindongwang/Deep-learning-activity-recognition`, 2018.

Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. Unsupervised data augmentation for consistency training. *Advances in neural information processing systems*, 33:6256–6268, 2020.

Peiyi Zhang, Xiaodong Jiang, Ginger M Holt, Nikolay Pavlovich Laptev, Caner Komurlu, Peng Gao, and Yang Yu. Self-supervised learning for fast and scalable time series hyper-parameter tuning. *arXiv preprint arXiv:2102.05740*, 2021.

Yong Zhang, Yu Zhang, Zhao Zhang, Jie Bao, and Yunpeng Song. Human activity recognition based on time series analysis using u-net. *arXiv preprint arXiv:1809.08113*, 2018.

A DATASET

Dataset	#Subjects	#Train	#Test
UCI	6	7352	2947
HAPT (imbalanced)	12	7767	3162
HAPT (balanced)	12	17076	3162
HHAR	6	155142	66491

Table 10: A brief description of three datasets.

A.1 UCIHAR DATASET

In the dataset, Y labels are represented as numbers from 1 to 6 as their identifiers:
- Walking as 1
- Waking_Upstairs 2
- Waking_Downstairs as 3
- Sitting as 4
- Standing as 5
- Laying as 6

All the data is present in the 'Dataset/UCI_HAR_dataset/' folder in the present working directory. Feature names are present in 'UCI_HAR_dataset/features.txt'.

Train Data:
- 'UCI_HAR_dataset/data/train/X_train.txt'
- 'UCI_HAR_dataset/data/train/subject_train.txt'
- 'UCI_HAR_dataset/data/train/y_train.txt'

Test Data:
- 'UCI_HAR_dataset/data/test/X_test.txt'
- 'UCI_HAR_dataset/data/test/subject_test.txt'
- 'UCI_HAR_dataset/data/test/y_test.txt'

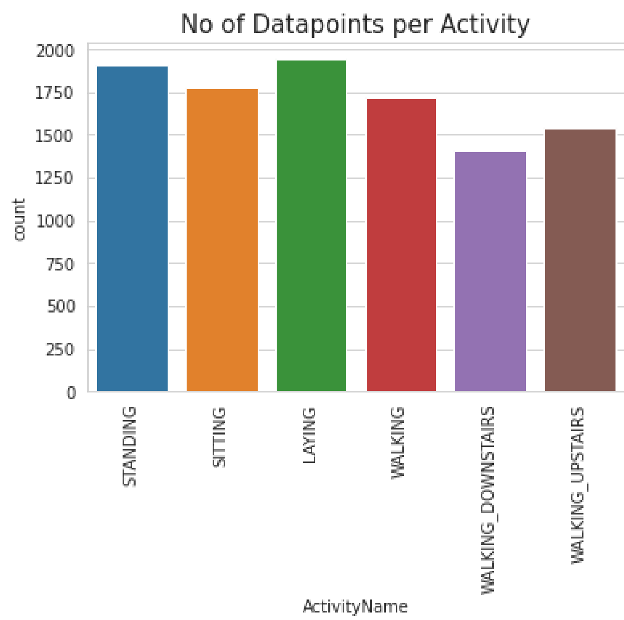

Figure 10: UCIHAR data count per activity.

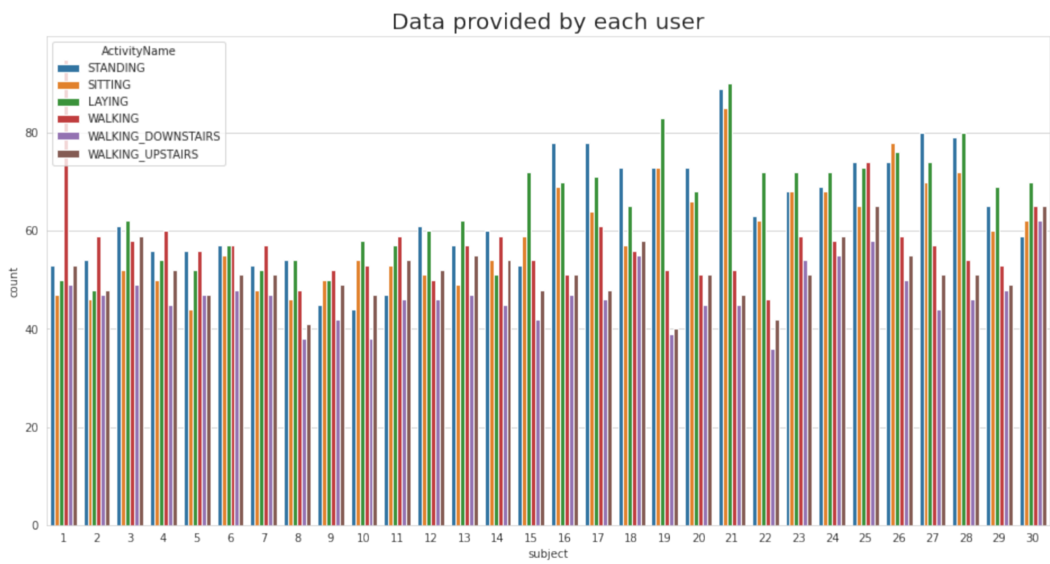

Figure 11: UCIHAR data provided by each user.

A.2 HAPT Dataset

In the dataset, Y labels are represented as numbers from 1 to 12 as their identifiers:
- Walking as 1
- Waking_Upstairs as 2
- Waking_Downstairs as 3
- Sitting as 4
- Standing as 5
- Laying as 6
- Stand_To_Sit as 7
- Sit_To_Stand as 8
- Sit_To_Lie as 9

- Lie_To_Sit as 10
- Stand_To_Lie as 11
- Lie_To_Stand as 12

All the data is present in the 'Dataset/HAPT Data Set' folder in the present working directory. Feature names are present in 'HAPT Data Set/features.txt'.

Train Data:
- 'HAPT Data Set/Train/X_train.txt'
- 'HAPT Data Set/Train/subject_id_train.txt'
- 'HAPT Data Set/Train/Train/y_train.txt'

Test Data:
- 'HAPT Data Set/Test/X_test.txt'
- 'HAPT Data Set/Test/subject_id_test.txt'
- 'HAPT Data Set/Test/y_test.txt'

It is highlighted that fewer numbers on the stand-to-sit, sit-to-stand, sit-to-lie, lie-to-sit, stand-to-lie, and lie-to-stand activities.

Oversample method

With Supratak (2020)'s 'get_balance_class_oversample(x, y)' function, we apply the oversample for HAPT dataset in data processing. First, it gets the class with the largest sample count. Second, in the imbalanced class, it randomly selects the samples to repeat until reaching the same number of samples as the largest.

A.3 HHAR Dataset

In the dataset, Y labels are represented as numbers from 1 to 6 as their identifiers:
- Biking as 1
- Sitting as 2

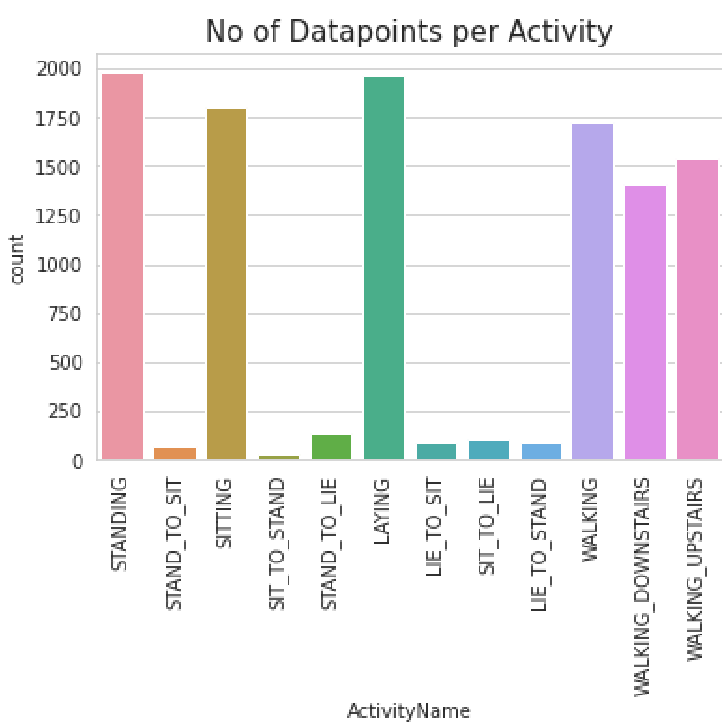

Figure 12: HAPT(imbalanced) data count per activity.

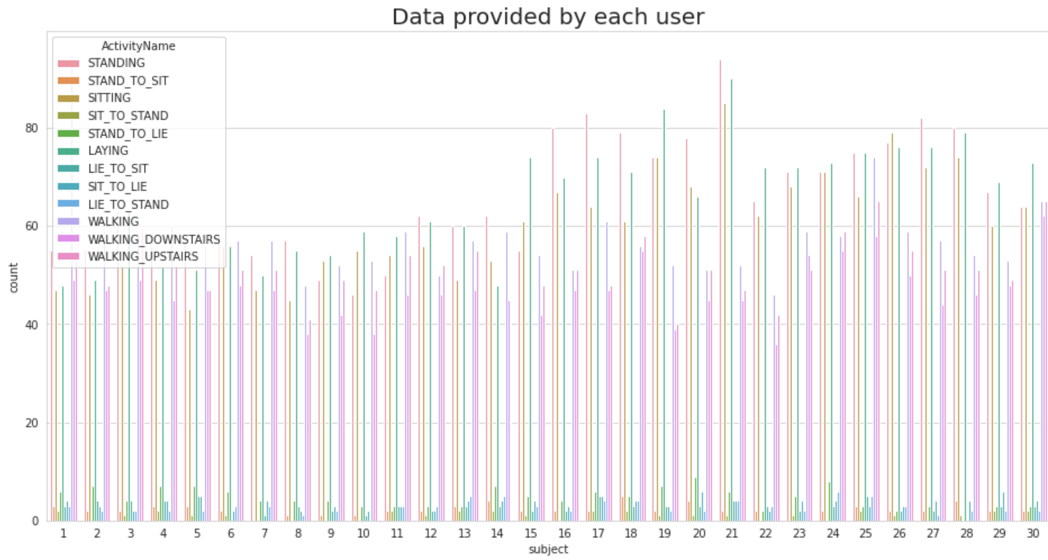

Figure 13: HAPT(imbalanced) data provided by each user.

- Stairs_Down 3
- Stairs_Up as 4
- Standing as 5
- Walking as 6

A.3.1 HHAR Dataset - Phone

All the data is present in the 'Dataset/HHAR Processed Data/HHAR_w' folder in the present working directory.

Train Data:
- 'HHAR_w/accelerometer/train.pt'
- 'HHAR_w/gyroscope/train.pt'

Test Data:
- 'HHAR_w/accelerometer/test.pt'
- 'HHAR_w/gyroscope/test.pt'

A.3.2 HHAR Dataset - Watch

All the data is present in the 'Dataset/HHAR Processed Data/HHAR_p' folder in the present working directory.

Train Data:
- 'HHAR_p/accelerometer/train.pt'
- 'HHAR_p/gyroscope/train.pt'

Test Data:
- 'HHAR_p/accelerometer/test.pt'
- 'HHAR_p/gyroscope/test.pt'

Tables 11 and 12 are the distribution of each activity with smartphones and smartwatches.

A.4 Data preprocess

With Wang (2018)'s work, we process the dataset from scratch.

Activity	Count of accelerometer	Count of gyroscope
Biking	1845557	1911730
Walking	2192401	2350429
Stairsdown	1615896	1673833
Stairsup	1782010	1884306
Standing	1851492	2024206
Sitting	1991919	2218501

Table 11: Count of activity of HHAR dataset-smartphone.

Activity	Count of accelerometer	Count of gyroscope
Biking	635530	522672
Walking	549761	488309
Stairsdown	486376	428241
Stairsup	473754	446023
Standing	451189	430223
Sitting	423995	419534

Table 12: Count of activity of HHAR dataset-smartwatch.

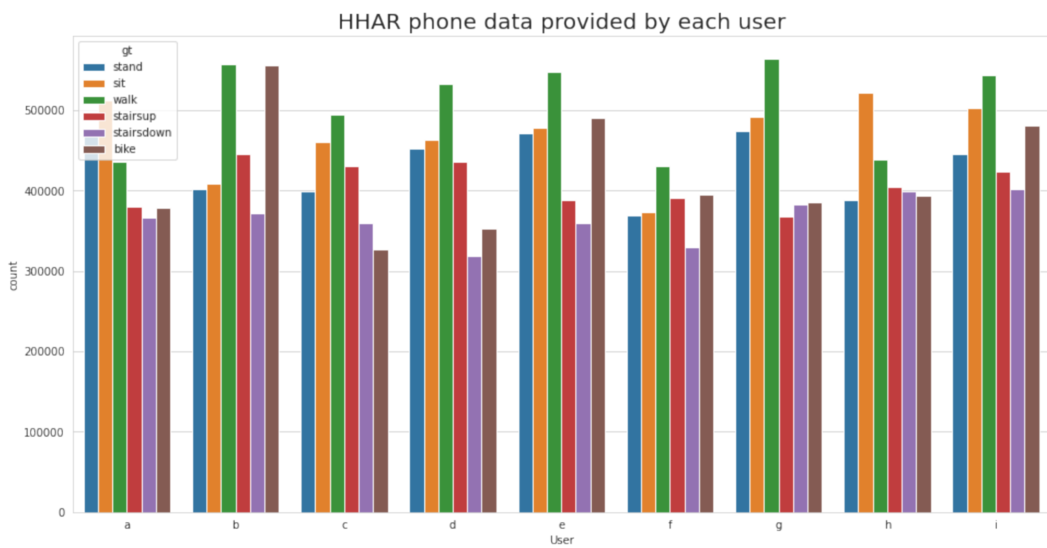

Figure 14: HHAR phone data provided by each user.

The sample X_train and X_test is parsed and reshaped into a 3-dimensional Numpy array in 'format_data_x(datafile)' function.

The function first initializes the x_data variable to None, and then loops through each item in the input datafile. For each item, it loads the data as a NumPy array with a float data type using np.loadtxt. If x_data is None, it initializes it as a NumPy array of zeros with the same length as item_data and a single column. It then horizontally stacks the item_data with the existing x_data. Finally, it removes the first column of x_data since it was added as zeros.

After formatting the data into a 2-dimensional array, the function then creates a new variable called X, which is initialized as None. It then loops through each row in the formatted x_data, reshapes it into a 9x128 array using np.asarray and row.reshape, and transposes it to get a 128x9 array. If X is None, it initializes it as a NumPy array of zeros with the same length as x_data, with each element being a 128x9 array. Finally, it sets the i-th element of X to be the reshaped and transposed row. The function then prints the shapes of the x_data and X arrays and returns X. The label Y_train and Y_test is parsed to one-hot encoded vector in 'format_data_y(datafile)' function.

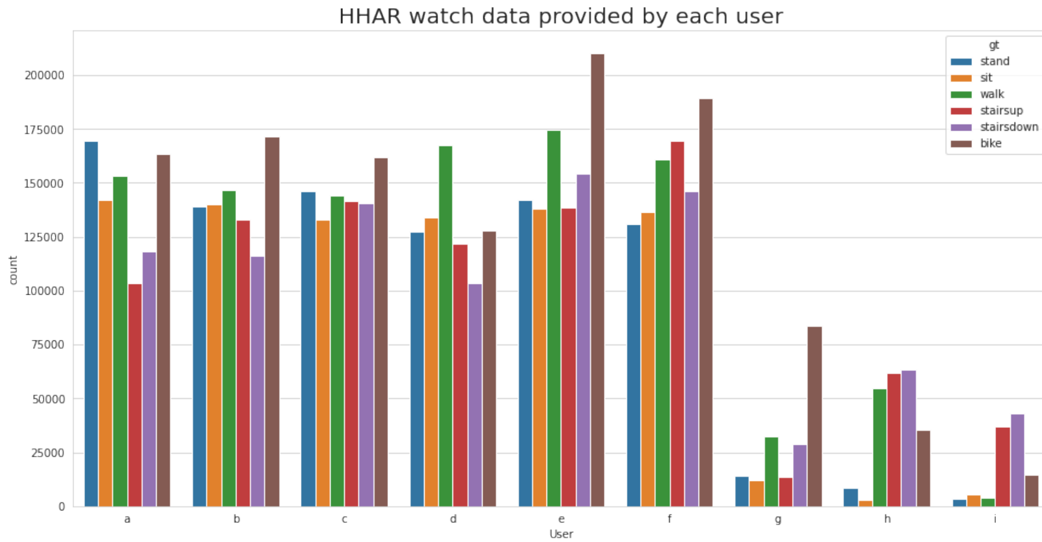

Figure 15: HHAR watch data provided by each user.

B NETWORK HYPERPARAMETERS

UCI				HAPT				HAHR			
CNN-LSTM		CNN-LSTM-ATT		CNN-LSTM		CNN-LSTM-ATT		CNN-LSTM		CNN-LSTM-ATT	
CNN1		**CNN1**		**CNN1**		**CNN1**		**CNN1**		**CNN1**	
In channels	9	In channels	9	In channels	3	In channels	3	In channels	3	In channels	3
out channels	64	out channels	64	out channels	64	out channels	64	out channels	64	out channels	64
kernel size	6	kernel size	6	kernel size	6	kernel size	6	kernel size	6	kernel size	6
stride	1	stride	1	stride	1	stride	1	stride	1	stride	1
padding	2	padding	2	padding	2	padding	2	padding	2	padding	2
ReLU()		**ReLU()**		**ReLU()**		**ReLU()**		**ReLU()**		**ReLU()**	
MaxPool1d		**MaxPool1d**		**MaxPool1d**		**MaxPool1d**		**MaxPool1d**		**MaxPool1d**	
kernel size	2	kernel size	2	kernel size	2	kernel size	2	kernel size	2	kernel size	2
CNN2		**CNN2**		**CNN2**		**CNN2**		**CNN2**		**CNN2**	
In channels	64	In channels	64	In channels	64	In channels	64	In channels	64	In channels	64
out channels	128	out channels	128	out channels	128	out channels	128	out channels	128	out channels	128
kernel size	3	kernel size	3	kernel size	3	kernel size	3	kernel size	3	kernel size	3
stride	1	stride	1	stride	1	stride	1	stride	1	stride	1
padding	2	padding	2	padding	2	padding	2	padding	2	padding	2
ReLU()		ReLU()		ReLU()		ReLU()		ReLU()		ReLU()	
MaxPool1d		**MaxPool1d**		**MaxPool1d**		**MaxPool1d**		**MaxPool1d**		**MaxPool1d**	
kernel size	2	kernel size	2	kernel size	2	kernel size	2	kernel size	2	kernel size	2
dropout()	0.1	dropout()	0.1	dropout()	0.1	dropout()	0.1	dropout()	0.1	dropout()	0.1
LSTM		**LSTM**		**LSTM**		**LSTM**		**LSTM**		**LSTM**	
Input size	64	Input size	64	Input size	47	Input size	47	Input size	32	Input size	32
hidden size	128	hidden size	128	hidden size	128	hidden size	128	hidden size	128	hidden size	128
num layers	1	num layers	1	num layers	1	num layers	1	num layers	1	num layers	1
tanh()		**tanh()**		**tanh()**		**tanh()**		**tanh()**		**tanh()**	
flatten()		**attention()**		**flatten()**		**attention()**		**flatten()**		**attention()**	
fc()		fc1()		fc()		fc1()		fc()		fc1()	
in features	128*128	hidden size	128	in features	128*128	hidden size	128	in features	128*128	hidden size	128
out features	6	fc2()		out features	12	fc2()		out features	6	fc2()	
softmax()		hidden size	128*2	**softmax()**		hidden size	128*2	**softmax()**		hidden size	128*2
dim	1	flatten()		dim	1	flatten()		dim	1	flatten()	
		fc()				**fc()**				**fc()**	
		in features	128			in features	128			in features	128
		out features	6			out features	12			out features	6
		softmax()				**softmax()**				**softmax()**	
		dim	1			dim	1			dim	1

Table 13: Supervised learning network hyperparameters set up on three datasets.

The function reads in data from a file using np.loadtxt, with a specified data type of np.int. The loaded data is then subtracted by 1 to ensure it starts at 0. np.eye(6) creates a 6x6 identity matrix, which is then indexed with the loaded data to create a one-hot encoded version of the data using np.eye. The resulting one-hot encoded data is returned as YY. After the data parsing, save them as train.pt, val.pt, and test.pt.

CNN SSL	
CNN1	**Value**
In channels	9 or 3
out channels	64
kernel size	6
stride	1
padding	2
ReLU()	
MaxPool1d	
kernel size	2
CNN2	
In channels	64
out channels	128
kernel size	3
stride	1
padding	2
ReLU()	
MaxPool1d	
kernel size	
CNN3	
In channels	128
out channels	256
kernel size	3
stride	1
padding	2
ReLU()	
MaxPool1d	
dropout	0.1
fc()	
in features	256
out features	number of the class. e.g.: 4 for SSL stage 1, 6 or 12 for SSL stage 2

Table 14: SSL network hyperparameters set up.

C MATERIAL RESOURCES AND COSTING

1. Anaconda: conda 4.14.0.
2. Python 3.9.12; pyTorch-nightly 1.13.0.
3. LucidChart: purchased LucidChart annual membership with FYP funding in USD $324.
4. Personal laptop

D PROJECT PLANNING

Regarding the final-year project planning, the timelines of each task are strictly followed. Throughout the entire project, I had the period review with supervisors to closely monitor the deliverable, which made sure to adhere strictly to the timeline. We frequently discussed the progress and addressed any potential issues that could lead to time deviation. The proactive approach toward project management and strict adherence to the timeline helped me complete the work without any delays or disruptions.

Phase 1 Project Proposal - Sem1		**13 January 2022**	**31 March 2022**
Initiative meet up	100%	13 January 2022	20 January 2022
Project introduction	100%	01 February 2022	31 March 2022
Literature review	100%	01 February 2022	31 March 2022
Project planning	100%	01 February 2022	27 February 2022
project budget	100%	01 February 2022	27 February 2022
Phase 2 Research and experiment on other's work - Sem1		**01 April 2022**	**30 July 2022**
Deep Learning tutorial	100%	01 April 2022	30 July 2022
Pytorch/Tensorflow tutorial	100%	01 April 2022	30 July 2022
Reseach on dataset and training model	100%	01 April 2022	30 July 2022
Experiment with current method	100%	01 April 2022	30 July 2022
'-Implement the current methods with dataset, reproduce the result			
Phase 3 Propose supervised learningn network with Dataset - Sem2		**01 August 2022**	**30 November 2022**
Pytorch tutrial	100%	01 August 2022	30 November 2022
My methodology with dataset	100%	01 August 2022	30 November 2022
Experiments and result	100%	01 August 2022	30 November 2022
Interim Report	100%	01 August 2022	30 November 2022
Phase 3 Propose supervised learningn network with Dataset - Sem3		**01 October 2022**	**31 March 2023**
My methodology with dataset	100%	01 October 2023	31 January 2023
Experiments and result	100%	01 February 2023	28 February 2023
Final report	100%	01 March 2023	20 March 2023

Figure 16: Project planning.