SELF-SUPERVISED LEARNING FRAMEWORKS FOR IOT APPLICATIONS

BY

DONGXIN LIU

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois Urbana-Champaign, 2022

Urbana, Illinois

ABSTRACT

Recent developments in deep learning have motivated the use of deep neural networks in Internet-of-Things (IoT) applications. But the performance of the deep neural network models for IoT applications, like those in other areas, largely depends on the availability of large labeled datasets, which in turn entails significant training costs. Such training costs mainly come from the burden of data labeling whereas the collection of the unlabeled data is a relatively easier process. Motivated by this observation, the self-supervised learning technique which could efficiently utilize the unlabeled data, has been widely studied and got great success in areas of computer vision and natural language processing (NLP). In IoT applications, however, self-supervised learning has not gotten enough attention. Considering the unique characteristics of IoT applications compared with computer vision or NLP tasks, customizing the self-supervised learning frameworks to IoT applications would be an interesting and important topic.

Different from computer vision or NLP applications, IoT applications often measure physical phenomena, where the underlying processes (such as acceleration, vibration, or wireless signal propagation) are fundamentally a function of signal frequencies and thus have sparser and more compact representations in the *frequency domain*. In this dissertation, we build self-supervised learning frameworks for IoT applications by carefully taking the frequency domain characteristics into consideration. We first studied the self-supervised contrastive learning framework, which demonstrated outstanding performance in areas of computer vision or NLP, on IoT applications from a time-domain perspective, and then re-designed the self-supervised contrastive learning framework from the frequency domain. After that, we designed two approaches to: 1) efficiently utilize both of the labeled and unlabeled data, and 2) reduce the dependency on the data augmentation strategies in the self-supervised learning frameworks. We evaluated the performance of our frameworks on public datasets with sensing data like radio signals, measurements of accelerometer and gyroscope, and WiFi signals. Next, we designed a seismic and acoustic signal based target detection system and deployed our self-supervised learning framework on it to study its performance on the real system. In the context of IoT, our customized self-supervised learning frameworks for IoT applications demonstrated obvious performance gain compared with the original ones designed for computer vision and NLP, which shows the effectiveness of our customization.

To my parents, for their love and support.

ACKNOWLEDGMENTS

First and foremost I am extremely grateful to my advisor, Prof. Tarek F. Abdelzaher for his invaluable advice, continuous support, and patience through the course of my Ph.D. study. His immense knowledge and plentiful experience have encouraged me in all the time of my academic research and daily life. I am always inspired by his passion, vision, and critical thinking for research.

I would also like to thank Prof. Matthew Caesar, Prof. Gang Wang, and Dr. Peng Wang for serving as my thesis committees, their insightful comments and feedbacks helped to improve my thesis to a higher level. Besides, I also thank Prof. Klara Nahrstedt, Prof. Lui Sha, and Prof. Indranil Gupta, for their kindly help and support during my Ph.D. study.

I would like to express special gratitude to Dr. Shuochao Yao for his invaluable help to this dissertation and my research. Additionally, I really enjoy working together with Dr. Yiran Zhao, Dr. Huajie Shao, Dr. Shengzhong Liu, Zhe Yang, Yifan Hao, Tai-Sheng Cheng, Jinyang Li, Tianshi Wang, Ruijie Wang, Yigong Hu, Jinning Li, Dachun Sun, and Chaoqi Yang. The experience of collaboration with them is impressive.

Finally, I would like to express my gratitude to my parents. Without their tremendous understanding and encouragement, it would be impossible for me to be a student in UIUC and complete my Ph.D. study.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION 1
- 1.1 Time-Domain Self-Supervised Contrastive Learning 3
- 1.2 Frequency-Domain Self-Supervised Contrastive Learning 4
- 1.3 Semi-Supervised Contrastive Learning 4
- 1.4 Spectrogram Masked Autoencoder for IoT Applications 5
- 1.5 A Case Study: Self-Supervised Learning on IoBT-OS 5
- 1.6 Dissertation Organization 6

CHAPTER 2 TIME-DOMAIN SELF-SUPERVISED CONTRASTIVE LEARNING FOR IOT 7
- 2.1 Overview 7
- 2.2 Related Work 8
- 2.3 Signal Model 10
- 2.4 Overview of SemiAMC 10
- 2.5 Self-Supervised Contrastive Pre-Training 11
- 2.6 Evaluation 13
- 2.7 Conclusion 18

CHAPTER 3 FREQUENCY-DOMAIN SELF-SUPERVISED CONTRASTIVE LEARNING FOR IOT 19
- 3.1 Overview 19
- 3.2 Related Work 20
- 3.3 Design of STFNet 23
- 3.4 Design of STF-CLS 34
- 3.5 Evaluation 39
- 3.6 Discussion and Limitations 45

CHAPTER 4 SEMI-SUPERVISED CONTRASTIVE LEARNING FOR IOT 48
- 4.1 Overview 48
- 4.2 Preliminary and Motivation 50
- 4.3 Design of SemiC-HAR 54
- 4.4 Evaluation 58
- 4.5 Conclusion 62

CHAPTER 5 SPECTROGRAM MASKED AUTOENCODER FOR IOT 64
- 5.1 Overview 64
- 5.2 Background and Related Works 66
- 5.3 Design of SMAE 68

5.4 Evaluation 73
5.5 Conclusion 78

CHAPTER 6 A CASE STUDY: SELF-SUPERVISED LEARNING ON IOBT-OS . 79
6.1 Overview 79
6.2 Background: The Decision Loop 80
6.3 IoBT-OS 81
6.4 The Case Study 82
6.5 Conclusion 86

CHAPTER 7 SUMMARY AND FUTURE WORK 87
7.1 Summary 87
7.2 Lessons 88
7.3 Future Work 89

REFERENCES 91

CHAPTER 1: INTRODUCTION

The Internet of Things (IoT) describes physical objects (or groups of such objects) that are embedded with sensors, processing ability, software, and other technologies that connect and exchange data with other devices and systems over the Internet or other communications networks[1]. Due to the development of ubiquitous computing, embedded systems, wireless sensor networks, control systems, and machine learning, the field of IoT has got great evolvement. Most of the current IoT applications measure physical phenomenas (e.g., acceleration, vibration, or wireless signal propagation) with corresponding sensors, and the measurements can be used for different tasks including health and wellness [1, 2, 3], behavior and activity recognition [4, 5, 6, 7], context sensing [8, 9, 10, 11], and object tracking and detection [12, 13, 14, 15, 16, 17].

Motivated by the increasing popularity of deep neural networks [18] that have demonstrated outstanding performance on various machine learning tasks, such as image classification [19, 20], object detection [21, 22], and natural language processing [23]. Deep learning techniques have also been widely studied in the field of IoT and got great success in IoT applications. Compared with the traditional approaches which designed machine learning models (such as SVM [24]) based on the manually designed features, the deep neural network models directly take the sensing signals as input and demonstrate much higher performance. Specific neural network models have been designed to fuse multiple sensory modalities and extract temporal relationships for sensing applications. These models have shown significant improvements on multiple IoT applications such as audio sensing[25, 26], tracking and localization [27, 28], and human activity recognition [29, 30].

Deep neural networks are traditionally trained using *supervised* learning algorithms that need a large amount of labeled training data. As deep learning techniques mature, the underlying neural network models often become deeper, thereby exacerbating the need for large-scale labeled datasets. Even when collecting the training data is not difficult, labeling or annotating the datasets usually requires extensive human labor work. As a result, self-supervised learning algorithms, which is a kind of unsupervised learning technique and build models only based on *unlabeled data*, have drawn more attention in recent years [31, 32, 33, 34, 35]. These techniques obviate extensive labeling burden, while attaining comparable performance with their supervised learning counterparts. In self-supervised learning scenarios, an encoder (that maps original inputs into lower-dimensional latent features or representations) is trained using *unlabeled data*. The learned latent feature can then

[1]`https://www.techtarget.com/iotagenda/definition/Internet-of-Things-IoT`

be further used to accomplish a variety of downstream tasks, such as classification or regression by training a much simpler model (*e.g.*, a linear model) that takes the latent feature (instead of the original data) as input. Only a small amount of labeled data is needed to train such (much simpler) models. As alluded to earlier, the big win lies in the fact that the bulk of the training (to generate the underlying latent representation) uses *unlabeled data*. Self-supervised learning has gotten outstanding performance in computer vision and NLP, but still not got a lot of attentions in IoT applications. Considering the unique frequency characteristics of sensing data for IoT applications, directly applying self-supervised learning framework from computer vision or NLP to IoT context would not get the best performance.

My dissertation works on customizing the self-supervised learning frameworks to IoT applications by carefully taking the frequency domain characteristics into consideration. The central design philosophy is inspired by the unique frequency domain characteristics that exist in the sensing signals of the IoT applications. IoT applications often measure physical phenomena, where the underlying processes (such as acceleration, vibration, or wireless signal propagation) are fundamentally a function of signal frequencies and thus have sparser and more compact representations in the frequency domain. We believe that designing the self-supervised learning frameworks for IoT applications from the perspective of frequency domain would largely improve their performance. A general workflow for self-supervised learning can be divided into two stages:

- **Pre-training:** Train the parameters of an encoder with large amount of unlabeled training data. Then freeze the encoder.

- **Downstream tasks:** The encoder trained in the pre-training stage would map the original inputs to their corresponding representations. A simple model can be built on the representations and then a small labeled dataset can be used to train the model.

Our customization would mainly focus on the pre-training stage. However, before the customization, we need to verify the feasibility of self-supervised learning frameworks on IoT applications. This means that we need to answer the question: "Does those frameworks designed for computer vision and NLP still work on dealing with sensing signals of the IoT applications?" Hence, we begin with studying the performance of self-supervised learning framework, which is a popular and widely-utilized self-supervised learning framework, on sensing signals, and then carried out the customizations from different aspects:

- **Frequency domain:** We designed the data augmentation (an important part of the self-supervised contrastive pre-training) and encoder from the frequency domain.

- **Labeled data:** We observed that the labeled data, even through with a small amount, can help to train a better encoder. And thus we proposed a semi-supervised (instead of self-supervised) contrastive pre-training framework.
- **Data Augmentation:** The performance of self-supervised contrastive learning largely rely on the choice of the data augmentation algorithm. A bad choice of data augmentation algorithm would reduce the quality of the trained encoder a lot. We thus designed a self-supervised learning framework for IoT application that does not rely on data augmentation.

In the following sections, we propose a brief introduction of how we customize the self-supervised learning frameworks to IoT applications.

1.1 TIME-DOMAIN SELF-SUPERVISED CONTRASTIVE LEARNING

In order to learn whether the self-supervised contrastive learning framework , which demonstrated outstanding performance in computer vision tasks, would perform well when dealing with sensing data, we directly applied the self-supervised contrastive learning framework to the sensing applications by designing time domain encoder and data augmentations, and then evaluated its performance. In this work, we built a *Semi*-supervised *A*utomatic *M*odulation *C*lassification framework, namely, *SemiAMC*, to efficiently utilize unlabeled radio signals. SemiAMC consists of two parts: (i) self-supervised contrastive pre-training and (ii) a downstream classifier. In self-supervised contrastive pre-training, we apply the design of SimCLR [34], which is an effective self-supervised contrastive learning framework, to train an encoder using a large amount of unlabeled training data. Then, we freeze the parameters of the encoder and train a classifier, which takes the representations (output of the encoder) as input, based on a small amount of labeled training data.

We evaluated the performance of SemiAMC on a widely-used modulation classification dataset RadioML2016.10a[36]. The evaluation results demonstrated that SemiAMC efficiently utilizes unlabeled training data to improve classification accuracy. Compared with previous supervised neural network models, our approach achieves better accuracies given the same number of labeled training samples. The evaluation results verified that the time domain approach do work on sensing data. In this work, we get a basic version of self-supervised contrastive learning framework for IoT applications, and in the next steps, we would show how our customization strategies help to improve the performance of it.

1.2 FREQUENCY-DOMAIN SELF-SUPERVISED CONTRASTIVE LEARNING

Existing self-supervised contrastive learning frameworks build their encoders with convolutional neural network (CNN) or recurrent neural network (RNN) which take the time domain data as input. However, our recent work has shown that for many IoT applications, the essential features of interest live in the frequency domain [37]. This insight calls for adapting the existing contrastive learning frameworks to frequency domain. In this part, we built the encoder from a time-frequency perspective using Short-Time Fourier Neural Networks (STFNet) [37] as basic building block. STFNet is a new neural network model specially designed for IoT applications. It operates directly in the frequency domain by mapping sensor measurements to the frequency domain with the Short-Time Fourier Transform (STFT), and was shown to perform much better than CNNs in many physical applications. We also design data augmentation operations from both time domain and frequency domain that enrich the contrastive prediction tasks for (a category of) IoT applications.

We evaluated the performance of our STFNet-based contrastive self-supervised learning framework on several human activity recognition (HAR) datasets, where the measurements of multiple sensors collected by different devices are used to determine the activities of a diverse group of participants. The experiments demonstrated obvious performance gains when we build the contrastive self-supervised learning framework from a time-frequency perspective instead of purely from the time-domain.

1.3 SEMI-SUPERVISED CONTRASTIVE LEARNING

From the label perspective, self-supervised learning is a special kind of unsupervised learning algorithm, which means that no label is required while training a model under self-supervised learning framework. A lot of NLP tasks are trained under this way. Not like NLP tasks, most IoT sensing tasks are classification or recognition tasks which do need labels. The labels here could provide extra information for the self-supervised contrastive learning frameworks. If we can find a way to make use of the labels during the self-supervised training, we would learn better latent features and hence improve the performance of the whole system.

To leverage this insight, we proposed *SemiC-HAR*, a semi-supervised contrastive learning framework for HAR. SemiC-HAR takes both labeled and unlabeled data into consideration. The key challenge of this work lies in how to efficiently utilize both labeled and unlabeled data in building the contrastive pre-training framework. To extract features from original inputs, previous research has studied self-supervised contrastive learning [34] which uses only unlabeled data and supervised contrastive learning [38] which uses only labeled data. In SemiC-HAR, we proposed a semi-supervised contrastive learning framework to efficiently utilize both labeled and unlabeled data in the pre-training process by carefully re-designing the contrastive loss function.

1.4 SPECTROGRAM MASKED AUTOENCODER FOR IOT APPLICATIONS

The performance of self-supervised contrastive learning highly relies on the choice of the data augmentation strategy. Different data augmentation strategies would lead to very different qualities of the trained encoder. To make things worse, different from computer vision (where the input is image or video) and NLP (where the input the natural language), the inputs for IoT applications are very different because different applications use different type sensors and thus yield different types of sensing signals, such as audio signals, seismic signals, radio signals, and measurements of motion sensors. Considering the difference of the physical phenomena behind the sensing tasks, the strategies to augment different sensing signals are very different. In this way, designing self-supervised contrastive learning frameworks requires a lot of human expertise on data augmentation. Hence, it would be very helpful if we could design a self-supervised learning framework for IoT applications that does not rely on data augmentation.

In this work, we proposed a new self-supervised learning framework based on the masked autoencoding approach [39], and re-designed the loss function to make it working in a better way to deal with spectrograms instead of images. The evaluation on both of the human activity recognition task and target detection task demonstrated the effectiveness of our customization.

1.5 A CASE STUDY: SELF-SUPERVISED LEARNING ON IOBT-OS

Finally, we studied the performance of our self-supervised learning framework on a real system instead of just running evaluations on different datasets. In this work, we built an operating system for the Internet of Battlefield Things (IoBT-OS) where our self-supervised learning framework plays an important role on improving decision accuracy and thus optimizing the latency of the decision loop. Then, we use a case study of seismic and acoustic based target detection to evaluate the performance of our proposed framework, and the evaluation result verified the effectiveness of our self-supervised learning framework.

1.6 DISSERTATION ORGANIZATION

The rest of this dissertation is organized as follows: In Chapter 2, we introduce the self-supervised contrastive learning framework, and show its performance on dealing with wireless radio signals from the time domain. We begin our customization by re-designing the data augmentation and structure of encoder from the perspective of frequency domain. This part of work is put in Chapter 3. After that, we propose our semi-supervised contrastive learning framework in Chapter 4. In Chapter 5, we show the work we have done to reduce the dependency on data augmentation. We then put the IoBT-OS case study in Chapter 6. Finally, we summarize this dissertation in Chapter 7.

CHAPTER 2: TIME-DOMAIN SELF-SUPERVISED CONTRASTIVE LEARNING FOR IOT

2.1 OVERVIEW

In this chapter, we use the modulation classification problem as an example to study the performance of the time-domain self-supervised contrastive learning framework. Automatically recognizing or classifying the radio modulation is a key step for many commercial and military applications, such as dynamic spectrum access, radio fault detection, and unauthorized signal detection in battlefield scenarios. The modulation classification problem has gotten widely studied in the past few years. Two general types of algorithms, likelihood-based (LB) and feature-based (FB), have been applied to solve the modulation classification problem. Likelihood-based methods [40, 41] make decisions based on the likelihood of the radio signal and achieve optimal performance in the Bayesian sense. However, they suffer from high computational complexity. Feature-based methods [42, 43, 44], on the other hand, make decision based on the manually-extracted features from the radio signal. Given the input radio signal, various features related to phase, amplitude or frequency are manually extracted and then used as inputs of the classification algorithms. Machine learning algorithms, such as support vector machine (SVM) and decision trees, are popular candidates for the classification algorithms.

Recently, with advances in deep learning techniques, deep neural networks achieves a great success in multiple fields, such as image classification [20] and natural language processing [23]. Deep neural network models, such as convolutional neural networks (CNNs) [45, 46] and recurrent neural networks (RNNs) [47, 48], have also been applied to the modulation classification problem. Such neural network models directly feed the raw signal data or its transforms as input and generally achieve much better performance than previous approaches.

However, training a deep neural network model requires a large amount of training data. In practice, it's usually difficult to collect a large volume of high quality and reliable radio signals as well as their modulations (labels) as training data. One common way to deal with the lack of training data is data augmentation. Previous studies proposed several data augmentation strategies [49, 50, 51] to avoid overfitting caused by the lack of training data. Another way is to utilize *unlabeled data*. The difficulty in collecting a large training dataset for radio modulation classification mainly comes from the labeling burden for radio signals. It is much eaiser to collect the radio signals without labeling them. Hence, it would help a lot if we could efficiently utilize *unlabeled* radio signals. As far as we know, the benefits of using unlabeled data have not yet been studied for the modulation classification problem.

Self-supervised learning, as a kind of unsupervised learning, obviates much of the labeling burden by extracting information from unlabeled data. Self-supervised learning algorithms have had great success in areas of computer vision [34], natural language processing [23], and IoT applications [30]. The goal of self-supervised learning is to train an encoder to extract useful intrinsic information (or features) from unlabeled data and use that information as the inputs to a classifier or predictor in a downstream task. Since these features already store a lot of intrinsic information about the original input, a simple classifier (*e.g.*, linear classifier) is enough for the downstream task. Using this approach, most training uses unlabeled data to learn the intrinsic features. Only a small amount of labeled data is then needed to train the downstream classifier.

In this chapter, we build a *Semi*-supervised *A*utomatic *M*odulation *C*lassification framework, namely, *SemiAMC*[52], to efficiently utilize unlabeled radio signals. SemiAMC consists of two parts: (i) self-supervised contrastive pre-training and (ii) a downstream classifier. In self-supervised contrastive pre-training, we apply the design of SimCLR [34], which is an effective self-supervised contrastive learning framework, to train an encoder using a large amount of unlabeled training data as well as data augmentation. Then, we freeze the parameters of the encoder and train a classifier, which takes the representations (output of the encoder) as input, based on a small amount of labeled training data.

We evaluated the performance of SemiAMC on a widely-used modulation classification dataset RadioML2016.10a[36]. The evaluation results demonstrated that SemiAMC efficiently utilizes unlabeled training data to improve classification accuracy. Compared with previous supervised neural network models, our approach achieved a better accuracy given the same number of labeled training samples.

The rest of this chapter is organized as follows. We introduce related work in Section 2.2. Section 2.3 is the signal model we used in this work. Section 2.4 and Section 2.5 describe the detailed design and implementation of SemiAMC. We cover the evaluation results in Section 2.6 and finally summarize this chapter in Section 2.7.

2.2 RELATED WORK

In this section, we briefly introduce related background on automatic modulation classification, self-supervised and semi-supervised learning techniques.

2.2.1 Deep Learning in Automatic Modulation Classification

Motivated by the success of deep learning techniques in computer vision [20] and NLP [23], deep neural network models, such as CNN [45], ResNet [46], and LSTM [47] have been applied to the automatic modulation classification task. Such neural network models directly take radio signals as input and predict the type of modulation as output. To further improve the performance of automatic modulation recognition, specific characteristics of the modulated radio signals are considered. For example, Zeng et al. [53] used spectrograms, generated from the radio signals through short-time discrete Fourier transform, as the input to a CNN classifier. Perenda et al. [54] separated the amplitude and phase series of the input and trained based on a parallel fusion method. Most of the previous works focused on designing supervised models while we design a semi-supervised learning framework.

2.2.2 Semi-Supervised Learning

Semi-supervised learning is a learning paradigm that leverages both labeled and unlabeled data to train machine learning models. With the development of deep learning techniques, deep neural network models become deeper and deeper and we need more and more labeled data to train such models. However, the collection of large datasets is very costly and time-consuming as it requires a lot of human labor work to annotate the dataset. In order to reduce the data labeling burden, a lot of semi-supervised learning algorithms [55] have been designed to make use of large number of unlabeled data as well as a small number of labeled data. In this work, our semi-supervised approach is based on self-supervised contrastive learning, which has gotten great success in computer vision area. To our best knowledge, we are the first who apply self-supervised contrastive learning to automatic modulation classification problem.

2.2.3 Self-Supervised Learning

Self-supervised learning is a learning paradigm where the model is trained on unlabeled data. Self-supervised contrastive learning is an typical self-supervised learning algorithm and has achieved outstanding performance [34]. The supervision of self-supervised contrastive learning comes from the user-designed pretext tasks which generate locally distorted data samples. A loss function can thus be generated that prefers mapping such similar inputs to nearby locations in the latent space.

Self-supervised learning can be used as the pre-training step in the semi-supervised learning algorithms [30, 34]. An encoder that collects intrinsic information from the original input (and maps it to a latent space) is trained with a self-supervised learning algorithm based on large amounts of unlabeled data. A small amount of labeled data can then be used to train a model on these features.

2.3 SIGNAL MODEL

We consider a single-input single-output communication system where we sample the in-phase and quadrature components of a radio signal through an analog to digital converter. The received radio signal $r(t)$ can be represented as

$$r(t) = c * s(t) + n(t), \tag{2.1}$$

where $s(t)$ refers to the modulated signal from the transmitter, c refers to the path loss or gain term on the signal, and $n(t)$ is the Gaussian noise. The received radio signal $r(t)$ is sampled N times at some sampling rate to obtain a length N vector of complex values. In this work, we treat the complex valued input as a 2-dimensional real-valued input. We denote it as X, where X is a $2 \times N$ matrix containing the in-phase (I) and quadrature (Q) components of the N received signal samples.

The goal of this chapter is to determine the modulation type for any given radio signal X.

2.4 OVERVIEW OF SEMIAMC

SemiAMC aims at training a classifier to accurately recognize the modulation type for any given radio signal. As a semi-supervised framework, SemiAMC is trained with both labeled and unlabeled data. We show the architecture of SemiAMC in Figure 2.1. The illustrated workflow is as follows.

The first step is called self-supervised contrastive pre-training, where we train an encoder to map the original radio measurements into low-dimensional representations. This is done in a self-supervised manner, with unlabeled data only. The supervision here comes from optimizing the *contrastive loss* function, that maximizes the agreement between the representations of differently augmented views for the same data sample.

In step two, we freeze the encoder learned during the self-supervised contrastive pre-training step, and map the labeled input radio signals to their corresponding representations in the low-dimensional space. The classifier can be trained based on these representations and their corresponding labels. Here a relatively simple classifier (*e.g.*, linear model) usually

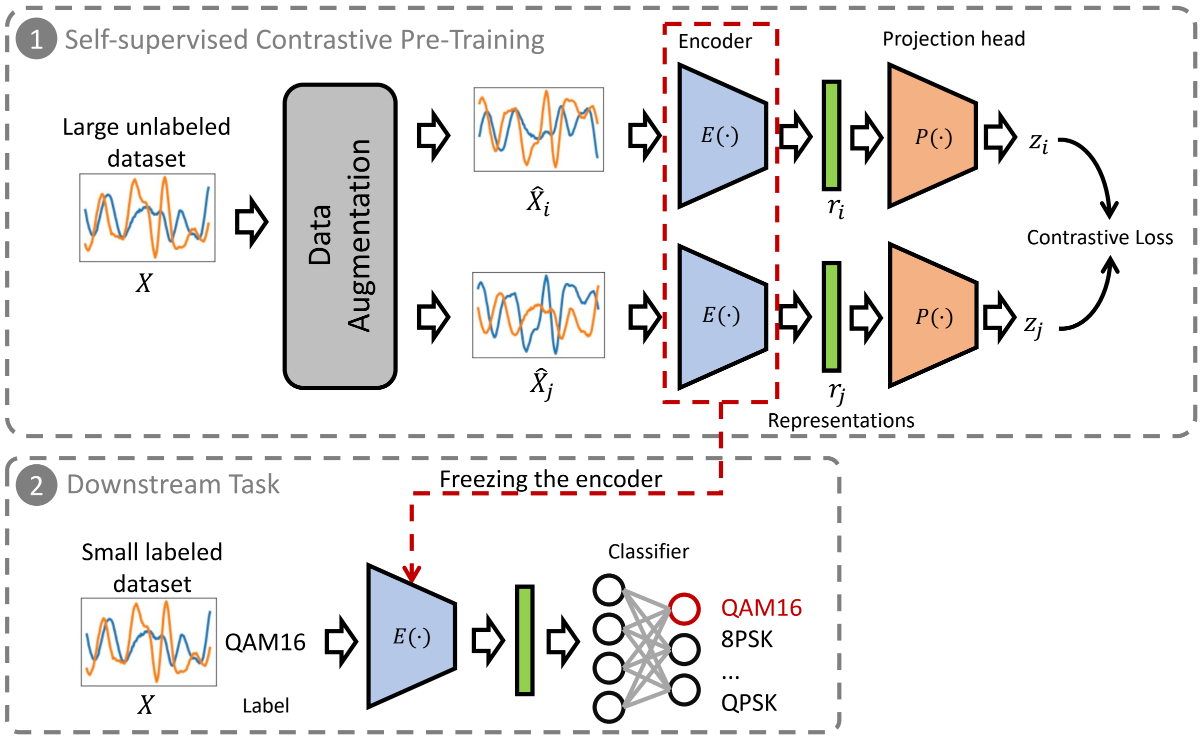

Figure 2.1: Overview of SemiAMC. We first train an encoder with a self-supervised contrastive learning framework using only unlabeled data. And then, the learned encoder and the corresponding features can be utilized to train the classifier with a small amount of labeled data.

work well, because the latent representation has already extracted the intrinsic information from the signal input. In this way, a small number of labeled data samples is enough to train the classifier. When we have enough labeled data, we can also fine-tune the last one or more layers of the encoder to further improve the performance of SemiAMC.

2.5 SELF-SUPERVISED CONTRASTIVE PRE-TRAINING

We applied SimCLR [34] as our self-supervised contrastive pre-training framework. As shown in Figure 2.1, our self-supervised contrastive pre-training mainly consists of four components.

2.5.1 Data Augmentation

The first component is a stochastic data augmentation module. Given any signal input X, two views of X, that are denoted as $\hat{X}_i$ and $\hat{X}_j$, are generated from the same family of data augmentation algorithms. The data augmentation algorithms are highly application dependent. For example, in image related applications, random color distortion, cropping, and Gaussian blur are commonly used data augmentation algorithms. In our approach, we augment the I/Q signal with the rotation operation [49] which could keep the features for classification. For a modulated signal $X = [I, Q]^T$, where I and Q refer to $1 \times N$ vectors storing the in-phase (I) and quadrature (Q) signals, we rotate it with an angle θ randomly selected from $\{0, \pi/2, \pi, 3\pi/2\}$. The augmented signal sample is

$$\hat{X} = \begin{bmatrix} \hat{I} \\ \hat{Q} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} I \\ Q \end{bmatrix}. \tag{2.2}$$

2.5.2 Encoder

The second part is a neural network based encoder E that extracts intrinsic information from the augmented examples $\hat{X}_i$ and $\hat{X}_j$, and stores them in the latent representations $\boldsymbol{r}_i$ and $\boldsymbol{r}_j$:

$$\boldsymbol{r}_i = E(\hat{X}_i), \ \boldsymbol{r}_j = E(\hat{X}_j). \tag{2.3}$$

Various choices of the network architectures can be used to design the encoder. For example, CNN-based encoders are widely utilized in learning representations of visual data and RNN-based encoders are usually utilized to deal with time-series. Figure 2.2 shows the architecture of the encoder we use in our approach. The I/Q signal input is first passed through a 1D convolutional layer (32 kernels with size 24) to extract the spatial characteristics. The following two LSTM layers (with 128 units) are used to extract the temporal characteristics. In the end, the 1D convolutional layer (128 kernels with size 8) and the max pooling layer are used to generate the representations.

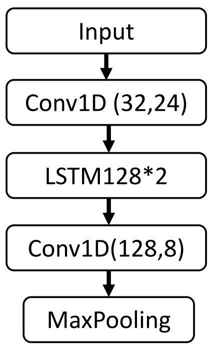

Figure 2.2: The architecture of the encoder.

2.5.3 Projection Head

The third part is a small neural network projection head P that maps representations to the space where contrastive loss is applied. In our approach, we use a multilayer perceptron (MLP) with one hidden layer as the projection head, which means

$$\boldsymbol{z}_i = P(\boldsymbol{r}_i) = W^{(2)}\sigma(W^{(1)}\boldsymbol{r}_i), \tag{2.4}$$

where σ is the ReLU activation function. It has been proved that calculating the contrastive loss on $\boldsymbol{z}_i'$s (instead of on the representations $\boldsymbol{r}_i's$ directly) improves the performance of self-supervised contrastive learning [34].

2.5.4 Contrastive Loss

The last part is the contrastive loss function that is defined for a contrastive prediction task aiming at maximizing the agreement between examples augmented from the same signal input. We use the normalized temperature-scaled cross entropy loss (NT-Xent) [56] as the loss function. For a given mini-batch of M examples in the training process, since for each example X, we will generate a pair of augmented examples, there will be $2M$ data points. The two augmented versions $\hat{X}_i$ and $\hat{X}_j$ of the same input X are called a positive pair. All remaining $2(M-1)$ in this batch are negative examples to them. Cosine similarity is utilized to measure the similarity between two augmented data samples. The similarity is calculated on $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$:

$$\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j) = \frac{\boldsymbol{z}_i^T \boldsymbol{z}_j}{\|\boldsymbol{z}_i\| \|\boldsymbol{z}_j\|}. \tag{2.5}$$

Here, $\|\boldsymbol{z}_i\|$ refers to the ℓ_2 norm of $\boldsymbol{z}_i$. The loss function for a positive pair of examples (i, j) is defined as

$$L_{i,j} = -\log \frac{\exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2M} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}, \tag{2.6}$$

where τ refers to the temperature parameter of softmax and $\mathbb{1}_{[k \neq i]} \in \{0, 1\}$ is an indicator function equaling to 1 iff $k \neq i$. The loss for all positive pairs (i, j) is calculated and the average is used as the final contrastive loss.

2.6 EVALUATION

In this section, we evaluated the performance of SemiAMC using a public dataset, RadioML2016.10a [36]. We first introduce the dataset we use and then show the experimental setup, including data preprocessing and detailed implementation. Finally, we analyze the performance of SemiAMC.

2.6.1 Dataset

We use RadioML2016.10a to study the performance of SemiAMC. RadioML2016.10a is a synthetic dataset including radio signals of different modulations at varying signal-to-noise ratios (SNRs). It consists of 11 commonly used modulations (8 digital and 3 analog): WBFM, AM-DSB, AM-SSB, BPSK, CPFSK, GFSK, 4-PAM, 16-QAM, 64-QAM, QPSK, and 8PSK. For each modulation, there are 20 different SNRs from -20dB to $+18$dB and there are 1000 signals under each SNR. Hence, the RadioML2016.10a has $1000 \times 20 \times 11 = 220,000$ signal examples in total. Each signal in RadioML2016.10a has 128 I/Q samples. In our approach, we put each signal in a 2×128 matrix X.

2.6.2 Experimental Setup

We split the dataset into three parts: training, validation, and testing by a ratio of 2:1:1. Specifically, for each modulation type and SNR, we randomly divide the 1000 signals into 500 signals for training, 250 signals for validation, and 250 signals for testing. Normalized signals are used as the input.

In the self-supervised contrastive pre-training part, we use a two-layer projection head. The first layer is a fully connected layer with 128 hidden units and ReLU activation. The second is a fully connected layer with 64 hidden units without an activation function. The encoder, the output of which is 128, is trained under a batch size of 512, with a total of 100 batches, and initial learning rate $1e^{-4}$ with cosine decay. In the downstream task part, we freeze the encoder and build a two-layer classifier on the representations. The first layer of the classifier is a fully connected layer with 128 neurons and ReLU activation. The second is a Softmax layer with 11 neurons, one for each modulation scheme. We apply droupout and L2 regularization to mitigate over-fitting.

We first study the performance when all the training and validation data have labels. We then study the performance when only part of the training and validation data have labels. We also study the performance of our approach when different amounts of unlabeled data are used to train the encoder. We further compute the classification accuracy separately for each SNR and modulation scheme. We run five times with different random seeds and take the average as the final performance.

2.6.3 Comparison with Supervised Frameworks

In this part, we compare the performance of our semi-supervised learning framework to that of previous supervised frameworks when we have enough labeled training data. Specifically, here we assume that we have labels for all the training and validation data. We first train the encoder with all signal samples in the training dataset. Then, we freeze the encoder and train the classifier based on both of the signals and labels in the training set. During this process, we also fine-tune the parameters of the encoder to get better result. We stop the training process when the validation loss does not decrease for 30 epochs and use the model with minimum validation loss to predict the classification accuracy on test set.

We compared the performance of SemiAMC against three supervised algorithms named CNN2 [45], ResNet [46], and LSTM2 [47]. As shown in Figure 2.3, SemiAMC performs the best even through there is no extra unlabeled data. The performance gain here mainly comes from the architecture of our encoder design and the data augmentation while training the encoder in self-supervised contrastive pre-training. SemiAMC clearly outperforms the supervised baselines above -2dB SNRs and achieves a maximum accuracy of 93.45% under 12dB SNR.

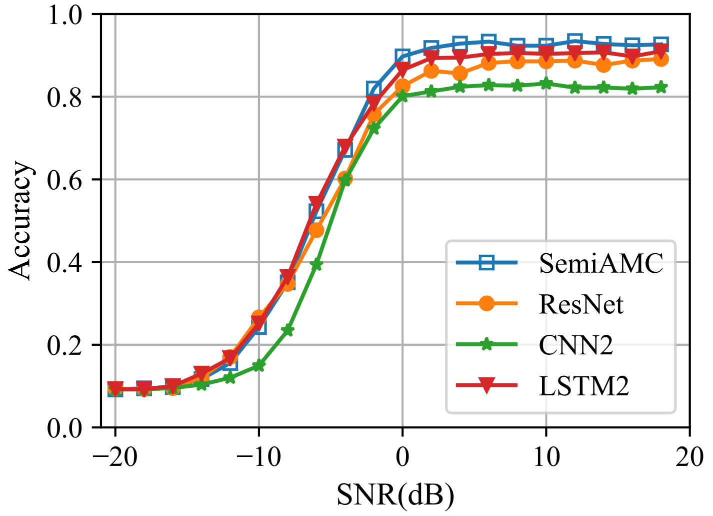

Figure 2.3: Comparison with other frameworks.

2.6.4 Performance under Different Amount of Labeled Data

In this part, we studied the performance of SemiAMC given a large amount of unlabeled data and a small amount of labeled data. For each modulation type under each SNR,

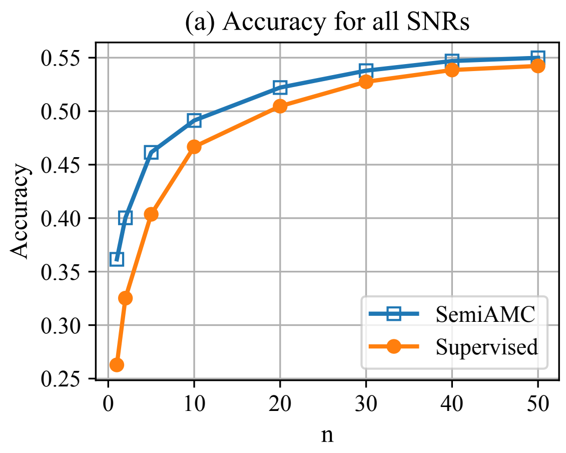

Figure 2.4: Accuracy under different amount of labeled data.

we have 500 signal samples for training, 250 signal samples for validation, and 250 signal samples for testing. We assume that for each modulation type under each SNR, only $n = 1, 2, 5(1\%), 10, 20, 30, 40, 50(10\%)$ signal samples have labels in the training set, and $\lceil n/2 \rceil$ signal samples have labels in the validation set. We train the encoder with all signal samples (without labels) in the training set, then we train the classifier and fine-tune the encoder with the labeled signal samples in the training set. The model with minimum validation loss is used to predict the classification accuracy on the test set.

To study the effectiveness of the encoder learned in the self-supervised contrastive pre-training, we directly train the encoder and classifier under a supervised way with the labeled data in the training set, and then use the model with minimum validation loss to recognize the signals in the test set. We denote this approach as "Supervised". The classification accuracy for test signals under all SNRs and SNRs larger than zero are shown in Figure 2.4. We can see that SemiAMC outperforms its corresponding supervised version. This confirms the effectiveness of the encoder trained with our self-supervised contrasive learning framework. We also observed that the gap between SemiAMC and the supervised version becomes larger with the decrease in labeled data. It suggests that our framework can effectively utilize unlabeled data. Naturally, the impact of unlabeled data is larger when less data are labeled. The performance gain of SemiAMC for SNRs larger than zero (Figure 2.4(b)) are larger than those computed across all SNRs (Figure 2.4(a)). It illustrated improved effectiveness of SemiAMC in dealing with higher SNR signals.

To further understand the classification accuracy under different SNR signals when different amounts of labeled data are given, we calculate the accuracy distribution under all SNR signals when n=1,5,10. The results are shown in Figure 2.5. We observed that the

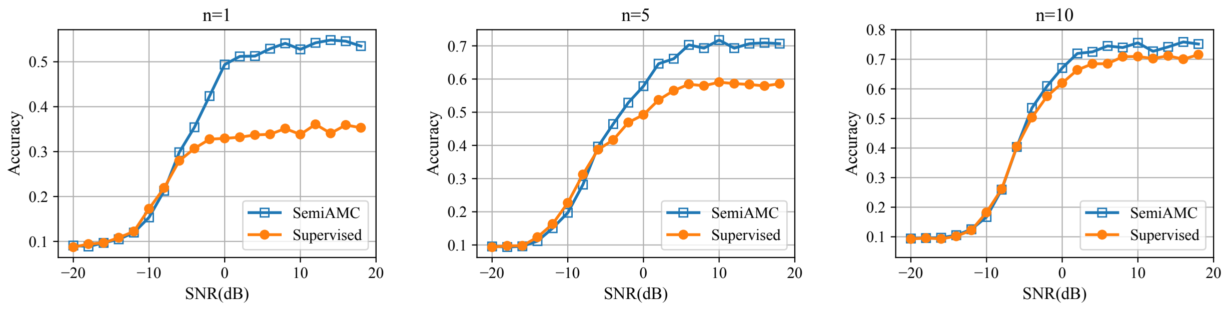

Figure 2.5: Accuracy for different SNRs when n=1,5,10.

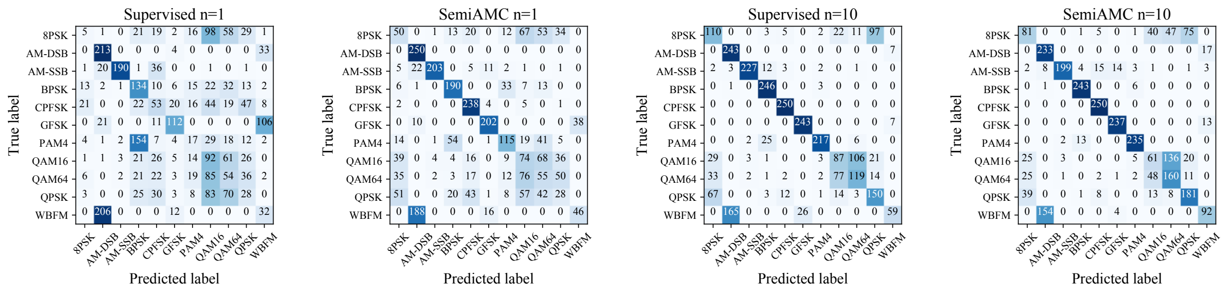

Figure 2.6: Confusion matrix of SemiAMC and Supervised when n=1,10.

performance gain of SemiAMC compared with the supervised approach mainly comes from the signals with high SNR while SemiAMC performs similarly with the supervised approach when the SNRs are small. We also studied the performance of SemiAMC of each modulation type when $n = 1, 10$ and the SNR is 10dB. We draw the corresponding confusion matrices in Figure 2.6. We observed similar accuracy patterns for different modulation types between SemiAMC and its corresponding supervised approach. For example, both of them would incorrectly recognize WBFM modulated signals as AM-DSB.

2.6.5 Performance under Different Amount of Unlabeled Data

Finally, we studied the recognition accuracy when different amounts of *unlabeled* data are given. Unlabeled data is used to train the encoder in the self-supervised pre-training part. In this experiment, we have 500 samples per SNR per modulation type for training. We assume that $n = 10$ of the samples have labels, and used $u = 0, 10, 20, 50, 100, 200, 300, 400, 490$ extra unlabeled samples besides the $n = 10$ labeled samples to train SemiAMC. Figure 2.7 plots the overall accuracy versus the change in the amount of unlabeled data. It is observed that the classification accuracy increases with increasing amounts of unlabeled data (up to a certain level).

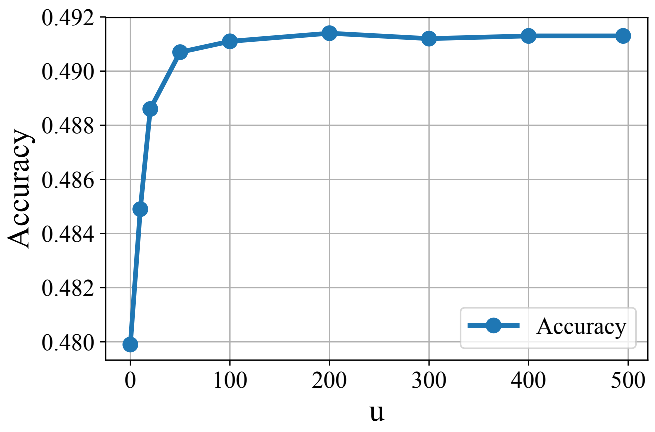

Figure 2.7: Accuracy under different amount unlabeled data.

2.7 CONCLUSION

In this chapter, we proposed a novel semi-supervised deep learning framework, called SemiAMC, to accurately recognize the modulation types of radio signals. SemiAMC efficiently utilizes unlabeled data by learning representations or features through self-supervised contrastive pre-training. We conduct several experiments on a public data set to evaluate the performance of SemiAMC. The evaluation results demonstrated a non-trivial performance gain for SemiAMC compared with supervised approaches for the same amount of labeled training data, thus verified the effectiveness of SemiAMC at utilizing unlabeled data.

CHAPTER 3: FREQUENCY-DOMAIN SELF-SUPERVISED CONTRASTIVE LEARNING FOR IOT

3.1 OVERVIEW

This chapter adapts the use of contrastive representation learning techniques to signals whose essential features are best represented in the *frequency domain*. Contrastive representation learning allows automatic extraction of essential features from complex signals by projecting them *in an unsupervised fashion* onto appropriate lower-dimensional spaces. Such projections can then be used for such purposes as signal classification, clustering, and entity detection. Prior work focused on time-domain signals. Our recent work has shown that, for many applications, the essential features of interest live in the frequency domain [37]. This insight calls for adapting the existing contrastive learning framework to frequency domain signals, which is the main contribution of this work.

This work is motivated by the increasing popularity of deep neural networks [18] that have demonstrated outstanding performance on various machine learning tasks, such as image classification [19, 20], natural language processing [23], and IoT applications [57, 58]. Deep neural networks are traditionally trained using *supervised* learning algorithms that need a large amount of annotated training data. As deep learning techniques mature, the underlying neural network models often become deeper, thereby exacerbating the need for large-scale annotated datasets. Even when collecting the training data is not difficult, annotating the datasets usually requires extensive human labor. As a result, self-supervised learning algorithms have drawn more attention in recent years [31, 32, 33, 34, 35]. These techniques obviate extensive labeling, while attaining comparable performance with their supervised learning counterparts. In self-supervised learning scenarios, an encoder (that maps original inputs into lower-dimensional latent representations) is trained using *unlabeled* data. The learned latent representation can then be further used to accomplish a variety of downstream tasks, such as classification or regression by training a much simpler model (e.g., a linear model) that takes the latent representation (instead of the original data) as input. Only a small amount of labeled data is needed to train such (much simpler) models. As alluded to earlier, the big win lies in the fact that the bulk of the training (to generate the underlying latent representation) uses *unlabeled data*.

The key to successful latent representation learning is to ensure that the learned latent representation captures the essential properties of the data, such as "similarity" in some appropriate semantic space. Teaching the encoder the right notion of similarity is often achieved by *data augmentation*, where input data samples are locally perturbed in a manner that does not substantially change their semantic interpretation. A cost function (used in training) ensures that such local perturbations are projected to *nearby locations* in the latent space. Designing the appropriate data perturbation for an application domain thus becomes an interesting research problem.

Recently, self-supervised learning (using data augmentation) has seen great success in computer vision [31, 33, 34, 59] and NLP [23, 35]. In IoT applications, however, the study of self-supervised learning has not gotten enough attention. Compared with computer vision or NLP tasks, the notion of "semantic similarity" in IoT applications is less clear. For example, in vision, an upside-down or mirror-image picture of a dog is still a dog, but what is a similarity notion in an accelerometer time-series or a backscatter signal received by a WiFi radio or RFID reader?

In this work, we apply the design of SimCLR [34], an effective framework for contrastive self-supervised representation learning (shown to perform well on the ImageNet dataset), but carefully redesign the structure of the encoder and the data augmentation operations. In lieu of using a convolutional neural network, we build the encoder using STFNet [37]. STFNet (or Short-Time Fourier Neural Networks) is a new neural network model specially designed for IoT applications. It operates directly in the frequency domain by mapping sensor measurements to the frequency domain with the Short-Time Fourier Transform (STFT), and was shown to perform much better than CNNs in many physical applications [37]. We also design data augmentation operations from both time domain and frequency domain that enrich the contrastive prediction tasks for (a category of) IoT applications.

We evaluate the performance of our STFNet-based contrastive self-supervised learning framework, we call STF-CSL, on several human activity recognition datasets, where the measurements of multiple sensors collected by different devices are used to determine the activities of a diverse group of participants. The experiments demonstrate obvious performance gains when we build the contrastive self-supervised learning framework from a time-frequency perspective (STF-CSL) instead of purely from time-domain (CNN-base approach).

The rest of this chapter is organized as follows. In Section 3.2, we introduce related work. Section 3.3 is the details of STFNet. We describe the architecture of our self-supervised learning model in Section 3.4. Section 3.5 covers the evaluation. Finally, Section 3.6 discusses the limitations of the approach.

3.2 RELATED WORK

In this section, we briefly overview related background on learning paradigms, IoT applications, self-supervision, and representation learning. These pieces, as we show later, serve as the building blocks of our contrastive self-supervised representation learning framework.

3.2.1 Deep Neural Network for IoT Applications

The impressive success in using deep neural networks for image classification [20] precipitated a reemergence of interest in deep learning. In recent years, deep neural networks have achieved significant performance improvements in multiple areas, including computer vision [19, 20], natural language processing [23, 60], and IoT applications [57, 58, 61].

In the context of IoT, researchers employed deep learning models to improve the predictive accuracy when time-series measurements (from sensors, such as accelerometers, gyroscopes, and magnetometers) generate inputs for various estimation and classification applications. Examples include applications in health and wellness [62, 63] and human activity recognition [64, 65, 66]. General frameworks were also proposed for integrating convolutional and recurrent neural networks for sensing applications [57]. To improve system efficiency at executing neural networks on low-end IoT devices, researchers have investigated means to compress structures and/or parameters of the neural network model, while keeping the accuracy almost the same [67, 68].

Inspired by the physics of measured processes, Yao et al. [37] addressed the customization of learning machinery to the frequency domain, and proposed a new foundational neural network building block, namely, the ShortTime Fourier Neural Network (STFNet). STFNet integrates neural networks with traditional time-frequency analysis, and performs the inner-layer operations in the frequency domain. STFNet outperformed previous CNN and/or RNN based approaches on a category of IoT applications. In this chapter, we build our self-supervised learning model based on STFNet, and evaluate its performance in self-supervised scenarios.

3.2.2 Self-supervised learning

Self-supervised learning, where the model is trained on unlabeled data, has become an increasingly popular framework for training deep neural networks. In a self-supervised learning framework, self-supervised tasks, also known as pretext tasks, are designed to generate locally perturbed data samples (with labels that capture the notion of local similarity in the perturbed unannotated data). A loss function can thus be generated that prefers mapping such similar inputs to nearby locations in the latent space.

Self-supervised learning has been widely studied in computer vision and NLP, and many ideas have been proposed to design the pretext tasks. For example, in computer vision, tasks such as predicting image rotations [31], solving jigsaw puzzles game [69], and colorful image colorization [32] are used to generate the self-supervision. The pretext tasks can also be implemented by a generative model. The generative modeling based self-supervised model can be trained to reconstruct the original input to learn the meaningful representations. For example, autoencoder [70] is trained to reconstruct the input images. The context encoder [33] is designed to recover the missing part of the input image when a mask is applied. Another commonly used framework in training self-supervised models is contrastive learning, where the task is designed to discriminate positive and negative samples. Chen et al. [34] designed a simple framework for contrastive learning of visual representations, namely, SimCLR and got impressive performance results on ImageNet. SimCLR learns representations for the input images by maximizing the agreement between the latent expression of two augmentations generated from the same input. Due to the simplicity and effectiveness of SimCLR, we apply it as our fundamental framework when customizing self-supervised learning framework to IoT applications.

In NLP research, self-supervised methods have also gotten widely studied. The original Word2Vec paper [71] proposed two architectures to compute continuous representations of words, where the pretext tasks are predicting the center word given the surrounding words, or, conversely, predicting the surrounding words given the center one. Devlin et al. proposed BERT [23] to pretrain deep bidirectional representations from unlabeled text by predicting randomly masked words in the text. It achieves excellent performance in language representation learning.

Customizing self-supervised learning models to the needs of IoT applications requires researchers to carefully design the encoder and data augmentation algorithms that work well for IoT applications. Existing work [4, 5, 6] implemented self-supervised learning frameworks for human activity recognition tasks. They designed convolutional or recurrent neural-network-based encoders that extract spatial and temporal properties of inputs, and applied data augmentation on the time-domain time-series inputs. Saeed et al. [4] proposed a multi-task self-supervised learning framework for sensory data and applied transformation discrimination as the pretext tasks. The model was trained to discriminate the transformed signals from the original inputs. Haresamudram et al. [6] introduced a masked reconstruction based self-supervised learning model for human activity recognition, where the self-supervision was generated by predicting the randomly masked timestep of the input sensor signal based on the remaining values. Tang et al. [5] explored the adoption and adaptation of SimCLR to human activity recognition, and designed a contrastive learning framework to learn the representations for sensor signal inputs. We advance this work by exploiting the conjecture that the underlying physics of measured phenomena in IoT applications are best expressed (e.g., are sparser and more compactly expressed) in the *frequency domain* [37], thus redesigning SimCLR to work with both time domain and frequency domain data. To make the scope of this work manageable, we focus on a subset of IoT applications exemplified by human activity recognition tasks.

3.2.3 Representation Learning

Representation learning, also called feature learning, is a way to automatically learn intrinsic structure and extract useful information from raw data that can effectively support impending machine learning tasks, such as regression and classification. The traditional manual feature engineering, which is labor-intensive, extracts or organizes useful information from data by relying on human ingenuity and prior knowledge. Representation learning [72, 73], however, allows the features to be learned by machines in an automatic way.

Principal component analysis (PCA) [74] and linear discriminant analysis (LDA) [75] are two early data representation learning algorithms. They learn low-dimensional representations of data with linear transformation techniques. In 2000, Roweis and Saul proposed locally linear embedding (LLE) [76], which is a nonlinear learning approach for generating low-dimensional neighbor-preserving representations from the high-dimensional input. With the development of deep learning, self-supervised neural networks have become widely utilized to extract the intrinsic representations in many domains, including images [32, 33, 34, 69, 77], video [78, 79, 80], audio [81, 82], and natural language processing [23, 83]. In this work, we show how self-supervised learning is customized to extract representations of sensor data by taking a frequency domain perspective.

3.3 DESIGN OF STFNET

In this section, we introduce the technical details of STFNets. We separate the technical descriptions into six parts. In the first two subsections, we provide some background followed by a high-level overview of STFNet components, including (i) hologram interleaving, (ii) STFNet-filtering, (iii) STFNet-convolution, and (iv) STFNetpooling. In the remaining four subsections, we describe the technical details of each of these components, respectively.

3.3.1 Background and STFNet Overview

IoT devices sample the physical environment generating time-series data. Discrete Fourier

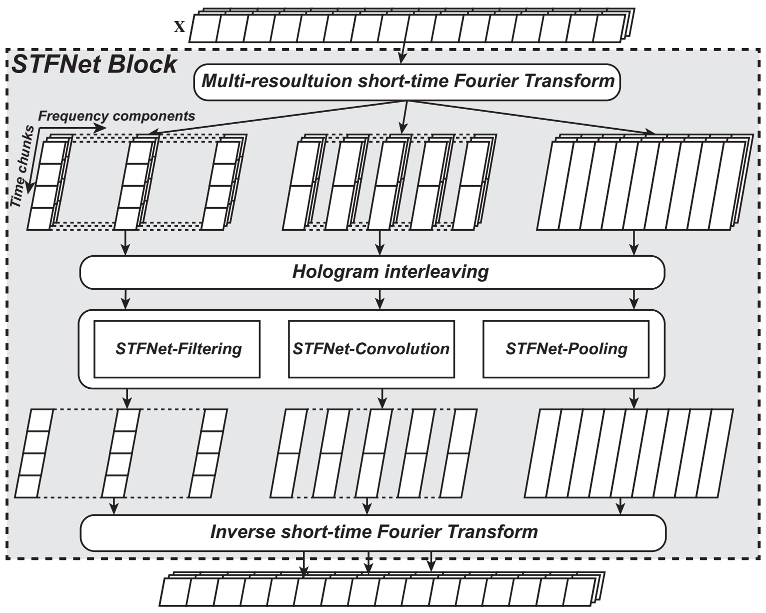

Figure 3.1: Overview design of STFNet block.

Transform (DFT) is a mathematical tool that converts n samples over time (with a sampling rate of f_s) into a n components in frequency (with a frequency step of f_s/n). The more samples are selected, the finer the component resolution is in frequency. We can always transform the whole sequence of data with DFT, achieving a high frequency resolution. However, we then lose information on signal evolution over time, or the time resolution. In order to solve this problem, Short-Time Fourier Transform (STFT) divides a longer time signal into shorter segments of equal length and computes DTF separately on each shorter segment. By losing a certain degree of frequency resolution, STFT helps us regain the time resolution to some extent. In choosing n, there arises a fundamental trade-off between the attainable time and frequency resolution, which is called the *uncertainty principle* [84]. For the purposes of learning to predict a given output, the optimal trade-off point depends on the time and frequency granularity of the features that best determine the outputs we want to reproduce. The goal of STFNets is thus to learn frequency domain features that predict the output, while at the same time learn the best resolution trade-off point in which the relevant features exist.

The building component of an STFNet is an *STFNet block*, shown in Figure 3.1. An STFNet block is the layer-equivalent in our neural network. The larger network would normally be composed by stacking such layers. Within each block, STFNet circumvents the uncertainty principle by computing multiple STFT representations with different time-frequency resolutions. Collectively, these representations constitute what we call the *time-frequency hologram*. And we call an individual time-frequency signal representation, a hologram representation. They are then used to mutually enhance each other by filling-in missing frequency components in each.

Candidate frequency-domain features are then extracted from these enhanced representations via general spectral manipulations that come in two flavors; filtering and convolution. They represent global and local feature extraction operations, respectively. The filtering and convolution kernels are learnable, making each STFNet layer a building block for spectral manipulation and learnable frequency domain feature extraction. In addition, there's a new mechanism, called pooling, for frequency domain dimensionality reduction in STFNets. Combinations of features extracted using the above manipulations then pass through activation functions and an inverse STFT transform to produce (filtered) outputs in the time domain. Stacking STFNet blocks has the effect of producing progressively sharper (i.e., higher order) filters to shape the frequency domain signal representation into more relevant and more fine-tuned features.

Figure 3.2 gives an example of an SFTNet block that accepts as input a two-dimensional time-series signal (e.g., 2D accelerometers data). Each dimension is then transformed to the frequency domain at four different resolutions using STFT, generating four different internal nodes, each of which representing the signal in the frequency domain at a different time-frequency resolution. Collectively, the four representations constitute the hologram. In the next step, mutual enhancements are done improving all representations. Each representation then undergoes a variety of alternative spectral manipulations (called "filters" in the figure). Two filters are shown in the figure for each dimension. The parameters of these filters are the weights multiplied by the frequency components of the filter input; a different weight per component. These parameters are what the network learns. Note that, a filter does not change the time-frequency resolution of the corresponding input. Filter outputs of the same time-frequency resolution are then combined additively across all dimensions and passed through a non-linear activation function (as in a conventional convolutional neural network). An inverse STFT brings each such combined output back to the time domain, where it becomes an input to the next STFNet block. (Alternatively, the inverse STFT can be applied after dimension combination and before the activation function.) Hence, each output time-series is produced by applying spectral manipulation and fusion to one particular time frequency resolution of all input time-series. Once converted to the time domain, however, the output time-series can be resampled in the next block at different time-frequency resolutions again. The goal of STFNet is to learn the weighting of different frequency components within each filter in each block such that features are produced that best predict final network outputs.

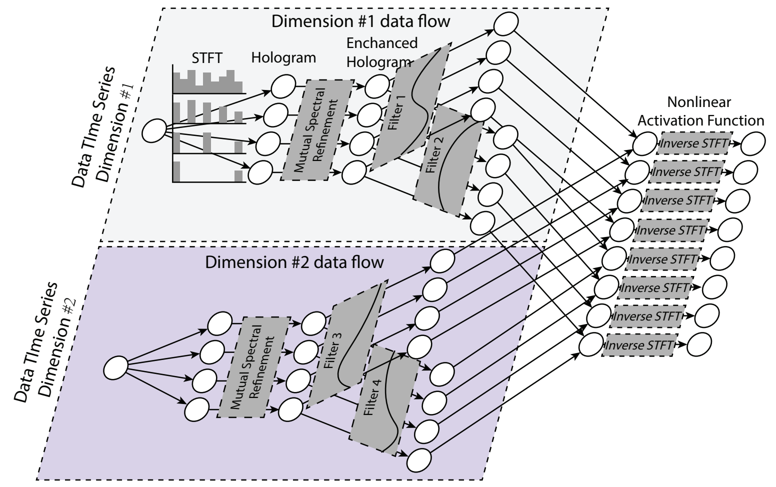
Dimension #1 data flow
Data TIme Series Dimension #1
STFT
Hologram
Enchanced Hologram
Mutual Spectral Refinement
Filter 1
Filter 2
Nonlinear Activation Function
Inverse STFT
Dimension #2 data flow
Data TIme Series Dimension #2
Mutual Spectral Refinement
Filter 3
Filter 4

Figure 3.2: Data Flow within a block of STFNet.

3.3.2 STFNet Block Fundamentals

In this subsection, we introduce the formulation of the design elements within each STFNet block. In the rest of this chapter, all vectors are denoted by lower-case letters (*e.g.*, $\mathbf{x}$ and $\mathbf{y}$), while matrices and tensors are represented by bold upper-case letters (*e.g.*, $\mathbf{X}$ and $\mathbf{Y}$). For a vector $\mathbf{x}$, the j^{th} element is denoted by $\mathbf{x}_{[j]}$. For a tensor $\mathbf{X}$, the t^{th} matrix along the third axis is denoted by $\mathbf{X}_{[\cdot,\cdot,t]}$, and other slicing denotations are defined similarly. We use calligraphic letters to denote sets (e.g., $\mathcal{X}$ and $\mathcal{Y}$). For set $\mathcal{X}$, $|\mathcal{X}|$ denotes the cardinality.

We denote the input to the STFNet layer as $\mathbf{X} \in \mathbb{R}^{T\times D}$, where the input D-dimension time-series are divided into windows of size T samples. We call T the signal length and D the signal dimension. Since STFNet concentrates on sensing signals, we assume that all the raw and internal-manipulated sensing signals are real-valued in time domain.

As shown in Figure 3.1, the input signal $\mathbf{X}$ first goes through a multi-resolution short-time Fourier transform (Multi_STFT), which is a compound traditional short-time Fourier transform (STFT), to provide a time-frequency hologram of the signal. STFT breaks the original signal up into chunks with a sliding window, where sliding window $\mathbf{W}(t)$ with width τ only has non-zero values for $1 \le t \le \tau$. Then each chunk is Discrete-Fourier transformed,

$$\mathbf{STFT}^{(\tau,s)}(\mathbf{X})_{[m,k,d]} = \sum_{t=1}^{T} \mathbf{X}_{[t,d]} \cdot \mathbf{W}(t - s\cdot m) \cdot e^{-j\frac{2\pi k}{\tau}(t-s\cdot m)}, \tag{3.1}$$

where $\mathbf{STFT}^{(\tau,s)}(\mathbf{X}) \in \mathbb{C}^{M\times K\times D}$ denotes the short-time Fourier transform with width τ and sliding step s. M denotes the number of time chunks. K denotes the number of frequency components. Since input signal $\mathbf{X}$ is real-valued, its discrete Fourier transform is conjugate symmetric. Therefore, we only need the $\lfloor \tau/2 \rfloor + 1$ frequency components to represent the signal, *i.e.*, $K = \lfloor \tau/2 \rfloor + 1$. STFNet uses sliding chunks with a rectangular window and no overlaps to simplify the formulation, *i.e.*, $s = \tau$ and $M = T/\tau$. Therefore the short-time Fourier transform can be denoted as $\mathbf{STFT}^{(\tau)}(\mathbf{X})$.

The Multi_STFT operation is composed of multiple short-time Fourier transform with different window widths $\mathcal{T} = \{\tau_i\}$. The window width, τ_i, determines the time-frequency resolution of STFT. Larger τ_i provides better frequency resolution, while smaller τ_i provides better time resolution. In STFNet, the window widths are set to be powers of 2, *i.e.*, $\tau_i = 2^{p_i}$ $\forall p_i \in \mathbb{Z}_0^+$. We can thus formulate Multi_STFT as:

$$\mathbf{Multi\_STFT}^{(\mathcal{T})}\{\mathbf{X}\} = \big\{\mathbf{STFT}^{(\tau_i)}(\mathbf{X})\big\} \text{ for } 2^{p_i} \in \mathcal{T}. \tag{3.2}$$

Next, according to Figure 3.1, the multi-resolution representations go into the hologram interleaving component, which enables the representations to compensate and balance their time-frequency resolutions with each other. The technical details of the hologram interleaving component are introduced in Section 3.3.3.

The STFNet layer then manipulates multiple hologram representations with the same set of spectral-compatible operation(s), including STFNet-filtering, STFNet-convolution, and STFNet-pooling. We will formulate these operations in Section 3.3.4, 3.3.5, and 3.3.6, respectively.

Finally, the STFNet layer converts the manipulated frequency representations back into the time domain with the inverse short-time Fourier transform. The resulting representations from different views of the hologram are weighted and merged as the input "signal" for the next layer. Since we merge the output representations from different views of the hologram, we reduce the output feature dimension of STFNet-filtering and convolution operations by the factor of $1/|\mathcal{T}|$ to prevent the dimension explosion.

3.3.3 STFNet Hologram Interleaving

In this subsection, we introduce the formulation of an innovative time-frequency domain operation proposed in STFNets: hologram interleaving. Due to the Fourier uncertainty principle, the representations in time-frequency hologram either have high time resolution or high frequency resolution. The hologram interleaving tries to use representations with high time resolution to instruct the representations with low time resolution to highlight the important components over time. This is done by two steps:

1. Revealing the mathematical relationship of aligned time-frequency components among

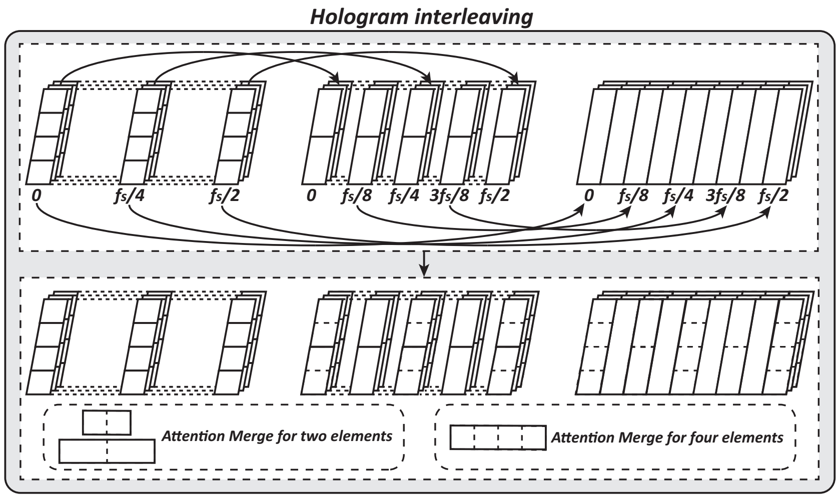

Figure 3.3: The design of hologram interleaving.

different representations in the time-frequency hologram.

2. Updating the original relationship in a data-driven manner through neural-network attention components.

We start from the definition of time-frequency hologram, generated by Multi_STFT defined in (3.2). Note that, the window width set $\mathcal{T}$ is defined as $\{2^{p_i}\}$, $\forall p_i \in \mathbf{Z}_0^+$. Without loss of generality, an illustration of multi-resolution short-time Fourier transformed representations with input signal having length 16 and signal dimension 3 as well as $\mathcal{T} = \{4, 8, 16\}$ are illustrated in Figure 3.3.

In order to find out the relationship of aligned time-frequency components, we start with the frequency-component dimension. Since different representations only change the window width τ_i of STFT but not the sampling frequency f_s of input signal, these frequency components represent frequencies from 0 to $f_s/2$ (Nyquist frequency) with step f_s/τ_i. Then we can first obtain the relationship of frequency ranging steps among different representations,

$$\forall p_i > p_j \ , \ \frac{f_s/\tau_j}{f_s/\tau_i} = 2^{p_i - p_j} \in \mathbf{Z}_0^+. \tag{3.3}$$

Therefore, a low frequency-resolution representation (with window width 2^{p_j}) can find their frequency-equivalent counterparts for every $2^{p_i - p_j}$ frequency components in a high frequency-resolution representation (with window width 2^{p_i}). The upper part of Figure 3.3 provides a simple illustration of such relationship. In the following analysis, we will use the original index k and corresponding frequency $k \cdot f_s/\tau_i$ interchangeably to recall the frequency component from the time-frequency hologram $\mathbf{STFT}^{(\tau)}(\mathbf{X})_{[m,k,d]}$.

Next, we analyze the relationship over the time-chunk dimension, when two representations have frequency-equivalent components. Note that time chunks in $\mathbf{STFT}^{(\tau)}(\mathbf{X})$ are generated by sliding rectangular window without overlap. Based on (3.1), for representations having window widths $\tau_i = 2^{p_i}$ and $\tau_j = 2^{p_j}$ $(p_i > p_j)$,

$$\begin{aligned}
\mathbf{STFT}^{(\tau_i)}(\mathbf{X})_{[m,2^{p_i-p_j}k,d]} &= \sum_{t=2^{p_i}m+1}^{2^{p_i}(m+1)} \mathbf{X}_{[t,d]} \cdot e^{-j\frac{2\pi 2^{p_i-p_j}k}{2^{p_i}}(t-m\cdot 2^{p_i})}, \\
&= \sum_{m_j=2^{p_i-p_j}m}^{2^{p_i-p_j}(m+1)-1} \sum_{t=m_j+1}^{2^{p_j}(m_j+1)} \mathbf{X}_{[t,d]} \cdot e^{-j\frac{2\pi k}{2^{p_j}}(t-m\cdot 2^{p_j})}, \\
&= \sum_{m_j=2^{p_i-p_j}m}^{2^{p_i-p_j}(m+1)-1} \mathbf{STFT}^{(\tau_j)}(\mathbf{X})_{[m_j,k,d]}.
\end{aligned} \tag{3.4}$$

Therefore, given the equivalent frequency component, a time component in low time-resolution representation (with window width 2^{p_i}) is the sum of $2^{p_i-p_j}$ aligned time components of the high time-resolution representation (with window width 2^{p_j}). As a toy example in Figure 3.3, the first row of the middle tensor is equal to the sum of first two rows of the left tensor for frequencies 0, $f_s/4$, and $f_s/2$. The row of the right tensor is equal to the sum of four rows of the left tensor for frequencies 0, $f_s/4$, and $f_s/2$. The row of the right tensor is equal to the sum of two rows of the middle tensor for frequencies $f_s/8$ and $3f_s/8$, etc.

According to the analysis above, the high frequency-resolution representations lose their fine-grained time resolutions at certain frequencies by summing the corresponding frequency components up over a range of time. However, the high time-resolution representations preserve these information.

The idea of hologram interleaving is to replace the sum operation in high frequency-resolution representation with a weighted merge operation to highlight the important information over time. For a certain frequency component, the weight of merging is learnt through the most fine-grained information preserved in the time-frequency hologram. In this work, we implement the weighted merge operation as a simple attention module. For a merging input $\mathbf{z} \in \mathbb{C}^{S\times 1}$, where S is the number of elements to be merged, the merge operation is formulated as:

$$\begin{aligned}
\mathbf{a} &= \mathrm{softmax}(|\mathbf{W}_m \mathbf{z}|), \\
y &= S \times \mathbf{a}^\intercal \mathbf{z},
\end{aligned} \tag{3.5}$$

where $|\cdot|$ is the piece-wise magnitude operation for complex-number vector; and $\mathbf{W}_m \in \mathbb{C}^{S\times S}$ is the learnable weight matrix. Notice that the final merged result is rescaled by the factor

S to imitate the "sum" property of Fourier transform.

3.3.4 STFNet-Filtering Operation

Spectral filtering is a widely-used operation in time-frequency analysis. The STFNet-filtering operation replaces the traditional manually designed spectral filter with a learnable weight that can update during the training process. Although the spectral filtering is equivalent to the time-domain convolution according to convolution theorem[1], the filtering operation helps to handle the multi-resolution time-frequency analysis, and facilitates the parameterization and modelling. We denote the input tensor as $\mathbf{X} \in \mathbf{C}^{M\times K\times D}$, where M is the number of time chunk, K is frequency component number, and D is input feature dimension. The STFNet-filtering operation is formulated as:

$$\mathbf{Y}_{[m,k,\cdot]} = \mathbf{X}_{[m,k,\cdot]}\mathbf{W}_{f[k,\cdot,\cdot]}, \tag{3.6}$$

where $\mathbf{W}_f \in \mathbb{C}^{K\times D\times O}$ is the learnable weight matrix, O the output feature dimension, and $\mathbf{Y} \in \mathbb{C}^{M\times K\times O}$ the output representation. The function of STFNet-filtering operation is providing a set of learnable global frequency template matchings over the time. However, it is not straightforward to extend the matching operation to the representations with different time-frequency resolutions. Although we can create multiple $\mathbf{W}_f$ with different frequency resolutions K, it can introduce unnecessary complexity and redundancy. STFNet-filtering solves this problem by interpolating the frequency components in weight matrix. As we mentioned in Section 3.3.3, data in hologram with different frequency resolutions have the same frequency range (from 0 to $f_s/2$) but different frequency steps (f_s/τ). Therefore, STFNet-filtering operation has only one weight matrix $\mathbf{W}_f$ with $K = \lfloor \tau/2 \rfloor + 1$ frequency components. When the operation input has $K' = \lfloor \tau'/2 \rfloor + 1$ frequency components with $K' < K$, we can subsample the frequency components in $\mathbf{W}_f$. When $K' > K$, we interpolate the frequency components of $\mathbf{W}_f$. STFNet provides two kind of interpolation methods: 1) linear interpolation and 2) spectral interpolation.

The linear interpolation generates the missing frequency components in extended weight matrix $\mathbf{W}'_f \in \mathbb{C}^{K'\times D\times O}$ from the two neighbouring frequency components in $\mathbf{W}_f$:

$$\begin{aligned} &k_l = \left\lfloor k'\frac{\tau}{\tau'} \right\rfloor \quad k_r = k_l + 1, \\ &\mathbf{W}'_{f[k',\cdot,\cdot]} = \mathbf{W}_{f[k_l,\cdot,\cdot]}\Big(k_r - k'\frac{\tau}{\tau'}\Big) + \mathbf{W}_{f[k_r,\cdot,\cdot]}\Big(k'\frac{\tau}{\tau'} - k_l\Big). \end{aligned} \tag{3.7}$$

[1]https://en.wikipedia.org/wiki/Convolution_theorem

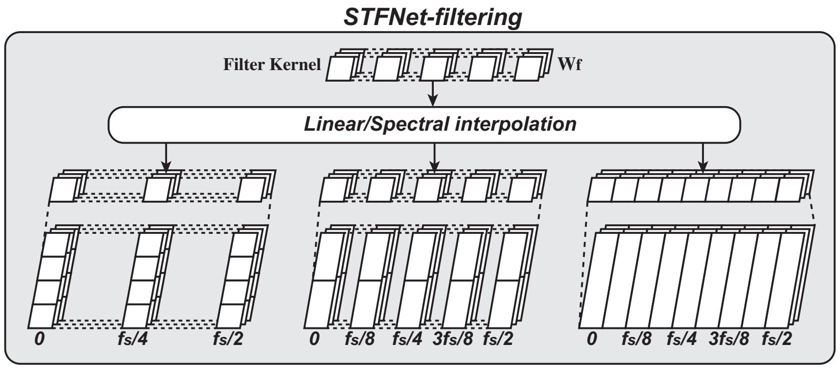

Figure 3.4: The STFNet-filtering operation.

The spectral interpolation utilizes the relationship between discrete-time Fourier transform (DTFT) and discrete Fourier transform (DFT). For a time-limited signal (with length τ), DTFT regards it as a infinite-length data with zeros outside the time-limited range, while DFT regards it as a τ-periodic data. As a result, DTFT generates a continuous function over the frequency domain, while DFT generates a discrete function. Therefore, DFT can be regarded as a sampling of DTFT with step f_s/τ. In order to increase the frequency resolution of $\mathbf{W}_f$, we can increase the sampling step from f_s/τ to f_s/τ', which is called spectral interpolation. Spectral interpolation can be done through zero padding in the time domain [84],

$$\mathbf{W}'_{f[\cdot,d,o]} = \mathbf{DFT}\Big(\mathbf{ZeroPad}_{\tau'-\tau}\mathbf{IDFT}\big(\mathbf{W}_{f[\cdot,d,o]}\big)\Big), \tag{3.8}$$

where $\mathbf{ZeroPad}_t$ denotes padding t zeros at the end of sequence, and $\mathbf{IDFT}(\cdot)$ denotes the inverse discrete Fourier transform. Please note that, if we pad infinite zeros to the IDFT result, then DFT turns into DTFT. An simple illustration of STFNet-filtering operation is shown in Figure 3.4.

3.3.5 STFNet-Convolution Operation

Other than the filtering operation that handles global pattern matching, we still need the convolution operation to deal with local motifs in the frequency domain. We denote the input tensor as $\mathbf{X} \in \mathbf{C}^{M \times K \times D}$, where M is the number of time chunk, K number of frequency component, and D input feature dimension. The convolution operation involves two steps: 1) padding the input data, and 2) convolving with kernel weight matrix $\mathbf{W}_c \in \mathbb{C}^{1 \times S \times D \times O}$, where S is the kernel size along the frequency axis and O is still the output feature dimension.

Without the padding step, the output of convolution operation will shrink the number

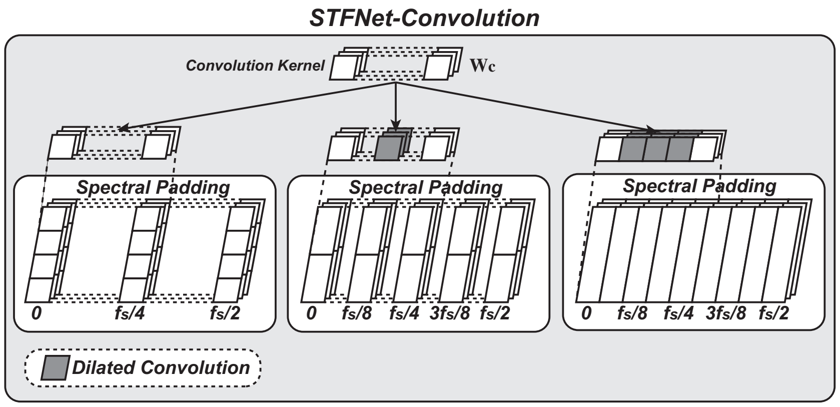

Figure 3.5: The STFNet-convolution operation with dilated configuration.

of frequency components, which may break the underlying structure and information in the frequency domain. Therefore, we need to pad extra "frequency component" to keep the shape of output tensor unchanged compared to that of the input data. In the deep learning research, padding zeros is a common practice. Zero padding is reasonable for inputs such as images and signal in the time domain, meaning no additional information in the padding range. However, padding zero-valued frequency component introduces additional information in the frequency domain.

Therefore, STFNet-convolution operation proposes the spectral padding for time-frequency analysis. According to the definition of DFT, transformed data is periodic within the frequency domain. In addition, if the original signal is real-valued, then the transformed data is conjugate symmetric within each period.

Previously, we cut the number of frequency components of a τ-length signal to $K = \lfloor \tau/2 \rfloor + 1$ for reducing the redundancy. In the spectral padding, we add these frequency components back according to the rule

$$\mathbf{X}_{[\cdot,\tau-k,\cdot]} = \mathbf{X}_{[\cdot,-k,\cdot]} = \mathbf{X}^*_{[\cdot,k,\cdot]}, \tag{3.9}$$

where $\mathbf{X}^*$ denotes complex conjugation. In addition, the number of padding before and after the input tensor is same as the previous padding techniques. Then we can define the basic convolution operation in STFNet

$$\mathbf{Y} = \mathbf{SpectralPad}(\mathbf{X}) \circledast \mathbf{W}_c, \tag{3.10}$$

where $\mathbf{SpectralPad}(\cdot)$ denotes our spectral padding operation, and $\circledast$ denotes the convolution operation.

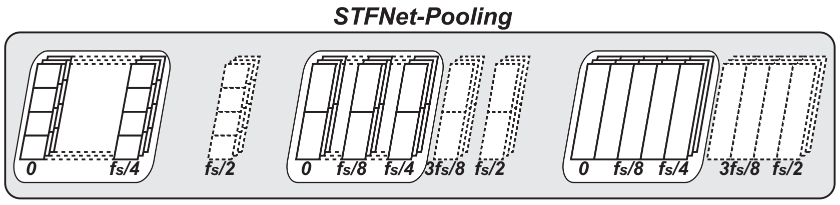

Figure 3.6: The low-pass STFNet-pooling operation.

Next, we discuss the way to share the kernel weight matrix $\mathbf{W}_c$ with multi-resolution data. Other than interpolating the kernel weight matrix as shown in (3.7) and (3.8), we propose another solution for the STFNet-convolution operation. The convolution operation concerns more about the pattern of relative positions on the frequency domain. Therefore, instead of providing additional kernel details on fine-grained frequency resolution, we can just ensure that the convolution kernel is applied with the same frequency spacing on representations with different frequency resolutions. Such idea can be implemented with the dilated convolution [85]. If $\mathbf{W}_c$ is applied to a input tensor with $K = \lfloor \tau/2 \rfloor + 1$ frequency components, for a input tensor with $K' = \lfloor \tau'/2 \rfloor + 1$ frequency components ($\tau' > \tau$), the dilated rate r is set to $\tau'/\tau - 1$.

$$r = \tau'/\tau - 1. \tag{3.11}$$

A simple illustration of STFNet-convolution with dilated configuration is shown in Figure 3.5.

3.3.6 STFNet-Pooling Operation

In order to provide a dimension reduction method for sensing series within STNet, we introduce the STFNet-pooling operation. STFNet-pooling truncates the spectral information over time with a pre-defined frequency pattern. As a widely-used processing technique, filtering zeroes unwanted frequency components in the signal. Various filtering techniques have been designed, including low-pass filtering, high-pass filtering, and band-pass filtering, which serve as templates for our STFNet-pooling. Instead of zeroing unwanted frequency components, STFNet-pooling removes unwanted components and then concatenates the left pieces. For applications with domain knowledge about signal-to-noise ratio over the frequency domain, specific pooling strategy can be designed. In this work, we focus on low-pass STFNet-pooling as an illustrative example.

To extend the STFNet-pooling operation to multiple resolutions and preserving spectral information, we make sure that all representations have the same cut-off frequency according to their own frequency resolutions. A simple example of low-pass STFNet-pooling operation is shown in Figure 3.6. We can see that our three tensors are truncated according to the same cut-off frequency, $f_s/4$.

3.4 DESIGN OF STF-CLS

Below, we first give an overview of our STFNet-based contrastive self-supervised representation learning framework, namely STF-CLS [30] , in section 3.4.1, and then provide the details of our design. We introduce the contrastive self-supervised learning framework (SimCLR) in section 3.4.2. Section 3.4.3 covers both of the time-domain and frequency-domain data augmentation operations. Finally, in section 3.4.4, we describe the design of STFNet and the corresponding encoders.

3.4.1 Overview

Our approach, STF-CSL, aims at learning good representations that store the intrinsic information of sensor measurement inputs in the latent space. The learned representations can be further utilized as the input of the target tasks such as classification or regression. Figure 3.7 presents an overview of STF-CSL. The illustrated workflow is as follows.

As a first step, called contrastive self-supervised pre-training, we train an encoder to map the original sensor measurements into a low-dimensional representation. This is done in a self-supervised manner, with unlabeled data only. Here, the supervision comes from optimizing a cost function, called *contrastive loss*, that aims at maximizing the agreement between the representations learned from locally perturbed data examples that are randomly generated by the same data augmentation operation.

We then freeze the parameters of the encoder learned during the contrastive self-supervised pre-training. Given any sensor measurement input, the encoder maps it to a corresponding low-dimensional representation. This representation facilitates downstream learning. A relatively simple model (*e.g.*, linear model) usually work wells, relating the latent representation to various desired outputs. Thus, we only need a small labeled training dataset to train the linear model. The last one or more layers of the encoder can also be fine-tuned during this supervised training process.

Next we show how to customizes this contrastive self-supervised representation learning framework to include frequency domain data.

3.4.2 Contrastive Self-Supervised Learning Framework

As shown in the contrastive self-supervised pre-training part of Figure 3.7, there are four

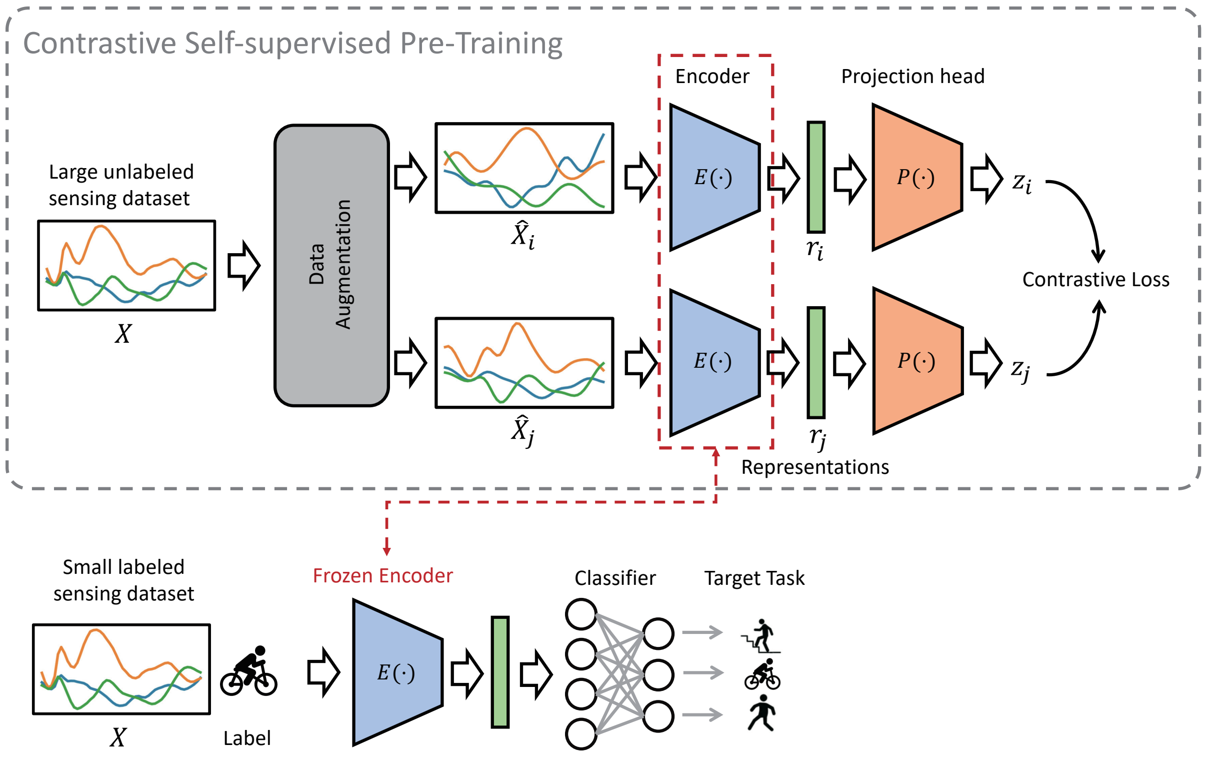

Figure 3.7: Overview of our designed STFNet-based contrastive self-supervised representation learning approach (STF-CSL). We train an STFNet-based encoder with a contrastive self-supervised learning framework using only unlabeled sensing data. And then, the learned encoder and the corresponding representations can be utilized to train the target model (*e.g.*, classification) with a small amount of labeled data.

major components.

The first part is a stochastic data augmentation module. Let X refers to the original input, which is the time-series measured from one or multiple sensors. $\hat{X}_i$ and $\hat{X}_j$ are two samples randomly generated from the same data augmentation family. Here, we define $\hat{X}_i$ and $\hat{X}_j$ as a positive pair of examples since they are generated from the same input X. In this work, we apply data augmentation operations from both of the time-domain and frequency-domain. The details are shown in section 3.4.3.

The second part is a neural network based encoder $E(\cdot)$ that maps the augmented example to the latent representations. Various choices of the network architectures can be used to design the encoder. For example, CNN-based encoders like ResNet [86] are widely utilized in learning representations of visual data. The encoder in our approach is built with STFNet and maps the data samples generated by data augmentation $\hat{X}_i$ and $\hat{X}_j$ to the corresponding representations $\mathbf{r}_i = E(\hat{X}_i)$ and $\mathbf{r}_j = E(\hat{X}_j)$.

The third part is a small neural network projection head $P(\cdot)$ that maps representations to the space where contrastive loss is applied. We use a multilayer perceptron (MLP) with one hidden layer to obtain $\mathbf{z}_i = P(\mathbf{r}_i) = W^{(2)}\sigma(W^{(1)}\mathbf{r}_i)$, where σ is ReLU activation function. The reason we apply the projection head before calculating the contrastive loss is that it has been proven that it is beneficial to define the contrastive loss on $\mathbf{z}_i$'s rather than $\mathbf{r}_i$'s [34].

The last part is a contrastive loss function defined for a contrastive prediction task. We use the normalized temperature-scaled cross entropy loss (NT-Xent) [56] as the loss function. For a given mini-batch of N examples in the training process. Since for each example X, we will generate a pair of augmented examples, and there will be $2N$ data points. The two augmented versions $\hat{X}_i$ and $\hat{X}_j$ of the same input X are called positive pair. All remaining $2(N-1)$ in this batch are negative examples. Cosine similarity is utilized to measure the similarity between two augmented examples $\hat{X}_i$ and $\hat{X}_j$. The similarity is calculated on $\mathbf{z}_i$ and $\mathbf{z}_j$:

$$\text{sim}(\mathbf{z}_i, \mathbf{z}_j) = \frac{\mathbf{z}_i^T \mathbf{z}_j}{\|\mathbf{z}_i\| \|\mathbf{z}_j\|}. \tag{3.12}$$

Here, $\|\mathbf{z}_i\|$ refers to the ℓ_2 norm of $\mathbf{z}_i$. The loss function for a positive pair of examples (i, j) is defined as

$$L_{i,j} = -\log \frac{\exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_k)/\tau)}, \tag{3.13}$$

where τ refers to the temperature parameter of softmax and $\mathbb{1}_{[k \neq i]} \in \{0, 1\}$ is an indicator function

$$\mathbb{1}_{[k \neq i]} = \begin{cases} 1 & \text{for} \quad k \neq i, \\ 0 & \text{for} \quad k = i. \end{cases} \tag{3.14}$$

The loss for all positive pair (i, j) would be calculated and the average of them is used as the final loss functions.

3.4.3 Data Augmentation

Data augmentation plays an important role in our contrastive self-supervised learning framework. Different data augmentation operations would lead to very different performances in learning the representations.

Existing work applies different time-domain data augmentation operations in their self-supervised learning frameworks for sensor measurements [4, 5, 6]. Frequency domain data augmentation, however, has not been studied in self-supervised learning although it was shown to be beneficial in a lot of time-series related applications such as time-series classification [87] and anomaly detection [88]. We apply both time-domain and frequency-domain data augmentation operations in our approach. Some data augmentation operations (*e.g.*, adding noise) can be used in many tasks, while others are more task-specific. In this work, we use several wearable device based human activity recognition datasets to demonstrate the effectiveness of our approach. Data augmentation operations are described below.

For the time-domain data augmentation, we use similar time-domain data augmentation operations with previous work [4, 5, 6].

- **Jitter**: Add random Gaussian noise to the input signals. This simulates the heterogeneity of the sensor hardware.
- **Rotation**: Apply a rotation matrix to the multi-dimensional input. Rotation of the input is related to differences in sensor placement with respect to measured objects, which is label-invariant.
- **Permutation**: This transformation first slices the input data into multiple segments along the time axis, and then permutes the segments to generate a new data example.
- **Time-warping**: This transformation generates a new data example with the same label by stretching and warping the time intervals between the values of the input time-series. Small changes on the temporal locations of the input would not change its label.
- **Scaling**: This transformation multiplies the data with a random scalar.
- **Inversion**: This transformation multiplies the input data with -1.
- **Horizontal flipping**: This transformation reverses the input data along the time-direction. Horizontally flipping generates a mirror version of the input data in the opposite time direction.
- **Channel-shuffling**: This transformation randomly permutes the channels of the multi-channel sensor input data.

Besides the time-domain data augmentation operations used by previous approaches, we also study the performance of the following three frequency-domain data augmentation operations in our contrastive self-supervised representation learning framework.

- **Low-pass filter**: This transformation is application specific, but for most physical systems with inertia, the key information in the sensor measurement time-series is stored in the low frequency part. This is because system inertia attenuates high-frequency signal components more so than low-frequency components. Thus, the presence of significant high-frequency components is often attributed to noise. The label of the input should therefore not change if we pass the input through a low-pass filter (which often increases signal-to-noise ratio). We randomly select the cutoff frequency in a reasonable range that is related to (and slightly higher than) the rate at which significant dynamics occur in the measured process.

- **Amplitude and phase perturbation**: For a given time series $\mathbf{x} = [x_1, x_2, ..., x_n]^T$, we first get its frequency domain expression by applying discrete Fourier transform on it

$$F(\omega_k) = \frac{1}{n}\sum_{t=1}^{n} x_t e^{-j\omega_k t} = \Re[F(\omega_k)] + j\Im[F(\omega_k)] = A(\omega_k)\exp[j\theta(\omega_k)] \tag{3.15}$$

  where $A(\omega_k)$ is the amplitude spectrum and $\theta(\omega_k)$ refers to the phase spectrum. In this transformation, the amplitude and/or the phase values of randomly selected segments of the frequency domain data are perturbed by Gaussian noise.

- **Phase shift**: Similar to phase perturbation, where the phase spectrum values are perturbed by Gaussian noise, the phase shift augmentation adds a random value between $-\pi$ and π to the phase values.

3.4.4 Design of the STFNet-based Encoder

Our encoder is based on the STFNets architecture for learning in the frequency domain [37]. An STFNet block is equivalent to one layer in CNNs, and the encoder in our approach is built by stacking multiple STFNet blocks. Each layer takes multi-dimensional time-series data as input, and computes multiple Short Term Fourier Transform (STFT) representations with different time-frequency resolutions that differ in the size of the input sliding window in the time domain and correspondingly the frequency resolution produced in the frequency domain. We set the multiresolution sliding window to $\{16, 32, 64, 128\}$. Frequency-domain features are then extracted through spectral manipulations (STFNet-Filtering, STFNet-Convolution, and STFNet-Pooling). The kernels of STFNet-Filtering and STFNet-Convolution are learnable. Our encoder uses three STFNet layers. Each STFNet layer is followed by a dropout layer with a keep-probability of 0.8. An 1D global average pooling layer is put to follow the last STFNet layer.

3.5 EVALUATION

In this section, we evaluate the STF-CSL system using several public datasets. We focus on device-based and device-free human activity recognition with motion sensors (accelerometer and gyroscope) and WiFi. We first introduce the datasets we use and then show the experimental setup, including data preprocessing and detailed implementation. Finally, we analyze the quality of representations studied by STF-CSL.

3.5.1 Datasets

We use human activity recognition data for the evaluation. We chose human activity recognition not because of its application value (it's an old application with existing off-the-shelf systems that already do very well), but rather because it is a widely-researched application with lots of available empirical data and many prior results that allow a comprehensive comparison of the proposed scheme to prior work. It allows us to demonstrate (using apples-to-apples comparisons) that extending contrastive learning to the frequency domain is a good idea. We use six human activity recognition datasets that cover a wide range of sensing signals (accelerometer, gyroscope, WiFi), devices (smartphone, smartwatch, WiFi transmitter and receiver), data collection protocols, and activity recognition tasks. Below, we introduce these datasets:

HHAR HHAR [89] is a human activity recognition dataset devised to benchmark human activity recognition algorithms in real-world context. The dataset is gathered with a variety of different device models including different smartphones and smartwatches. In this dataset, a total of six different activities are studied: biking, sitting, standing, walking, upstairs and downstairs. Four smartwatches (2 LG watches, 2 Samsung Galaxy Gears) and eight smartphones (2 Samsung Galaxy S3 mini, 2 Samsung Galaxy S3, 2 LG Nexus 4, 2 Samsung Galaxy S+) are used by 9 users to record accelerometer and gyroscope readings under different activities. The sampling rate of signals varied across phones and watches with values between 50-200Hz.

MobiAct MobiAct [90] is a benchmark dataset for smartphone-based human activity recognition. It collects signals from montion sensors (3D accelerometer, gyroscope, orientation) in a Samsung Galaxy S3 smartphone to perform fall detection and recognize activities of daily living. It records 4 types of falls and 12 types of activities of daily living (ADL) from a total of 66 participants of different age, height and weight. More than 3200 trials

Table 3.1: A summary of the datasets

Dataset	No. of Users	No. of Activities
HHAR	9	6
MobiAct	66	10
MotionSense	24	6
UCI HAR	30	6
WISDM	29	6
EI	11	6

were collected, all captured by the same Samsung Galaxy S3 smartphone. The smartphone was freely placed in the participants' pockets with random orientation. In our evaluation, we use the data related to 10 activities to evaluate the performance of our approach. The considered activities are: standing (STD), walking (WAL), jogging (JOG), jumping (JUM), stairs up (STU), stairs down (STN), standing to sitting on a chair (SCH), sitting on a chair (SIT), car-step in (CSI) and car-step out (CSO).

MotionSense MotionSense dataset [91] includes time-series data generated by accelerometer and gyroscope sensors. It is collected with an iphone 6, kept in the participants's front pocket. A total of 24 participants in a range of gender, age, weight, and height perform six activities, including downstairs, upstairs, walking, jogging, sitting, and standing in the same environment and condition.

UCI HAR UCI HAR [92] is a public dataset which studies human activity recognition using smartphones. A total of 30 participants with ages from 19 to 48 years old were selected in this dataset. Each participant wears a Samsung Galaxy S II smartphone on their waist. Six activities were performed by the participant: standing, sitting, laying down, walking, walking downstairs and walking upstairs. The readings of the accelerometer and gyroscope were collected at a sampling rate 50Hz.

WISDM WISDM [93] dataset is released by the Wireless Sensor Data Mining Lab. It's an early dataset which studies human activity prediction with mobile devices. It contains data collected from 29 participants. The participants take a smartphone including Nexus One, HTC Hero, and Motorola Backflip and perform six different activities: walking, upstairs, jogging, downstairs, sitting, and standing. The outputs of the accelerometers are collected in a 20Hz sampling rate.

EI EI [94] is a deep learning based device-free human activity recognition framework. It made use of Channel State Information (CSI) to analyze human activity. The CSI data of 11 participants with different ages, height, and weight was collected from 6 different rooms in two buildings. The CSI values of 30 OFDM subcarriers were recorded to analyze six different types of human activities: wiping the whiteboard, walking, moving a suitcase, rotating the chair, sitting, as well as standing up and sitting down.

We summarize the information of the datasets in Table 3.1.

3.5.2 Experiment Setup

In this part, we first introduce how we preprocess the datasets. Next, we show the detailed implementation of our approach.

Data Preprocessing For HHAR, MobiAct, MotionSense, UCI HAR, and WISDM, we use the accelerometer data. We use all the measured CSI when considering the EI dataset. We linearly interpolate the reading to 50Hz and then segment the datasets into sliding windows of size 512 with 50% overlap. For each dataset, we use the data from approximately 75% of users as the training data and the remaining as the test set.

Implementation Details We follow the general architecture in Figure 3.7. The encoder consists of three STFNet layers with $\{64, 64, 256\}$ kernels. We set the sliding window for multiresolution short-time Fourier transform to be $\{16, 32, 64, 128\}$ for all the three layers. Each STFNet layer is followed by a dropout layer with keep probability 0.8. A global average pooling layer is put in the end of the encoder. We utilize a two-layer projection head. The first layer is a fully connected layer with 256 hidden units and ReLU activation function. The second is a fully connected layer with 64 hidden units without activation function. We apply a combination of rotation and amplitude and phase perturbation as our default data augmentation operation on HHAR, MobiAct, MotionSense, UCI HAR, and WISDM dataset. For EI, we use the amplitude and phase perturbation as augmentation operation.

Our model was trained using an Adam optimizer [95] with learning rate 10^{-4} and decay rate 0.9 and 0.99 for the first and second moment, respectively. We add L2 regularization to the loss function in order to mitigate over-fitting. We use a batch size of 64, and train the model for 50 epochs. In the evaluation, we run 5 times for each experiment and take the average as the final performance.

Linear Classification To evaluate the quality of the representations learned by our approach, we build a linear layer on the representations to do the classifications. After the contrastive self-supervised pre-training, we freeze the parameters of the learned encoder, and train the linear classifier to implement human activity recognition.

Models in Comparison In this evaluation, we are trying to show the superiority of STFNet, compared with CNN, on contrastive self-supervised learning framework. Hence, we build a three-layer CNN followed by a global average pooling layer. The CNN based encoder has the same structure with our encoder. We summarize the models we used in evaluation as follows:

- **STF-CSL:** In this model, we use our STFNet-based encoder in the contrastive self-supervised pre-training and then freeze the parameters of the encoder. A linear layer built on the output of the encoder, which means the representations we learned, is used to do the classification. The labeled dataset is used to train the linear layer.
- **CNN-CSL:** This model uses CNN instead of STFNet to build the encoder. The remaining parts are the same with STF-CSL. Tang et al. also applied this approach in their work [5].
- **Supervised:** We compare with performance of our approach with supervised model. In this way, we train the STFNet-based encoder and the linear layer directly in a supervised way.

3.5.3 Results

We show the evaluation results in this part. We begin with the comparison between CNN and STFNet, and then study the performance under different data augmentation operations. At last, we show the visualization of the learned representations.

Comparison between CNN and STFNet We first compare the performance of CNN and STFNet in the contrastive self-supervised learning framework. We train the CNN-based encoder (CNN-CSL) and the STFNet-based encoder (STF-CSL) with the whole training set (without labels) in a self-supervised way. And then we train the linear classifier when $1\%, 2\%, 5\%, 10\%, 20\%, 50\%$ and 100% labels are given, and calculate the corresponding F1 scores on the test sets. The results are shown in Figure 3.8. We can observe that STF-CSL (orange curves) obviously outperforms CNN-CSL (green curves) on different datasets. This

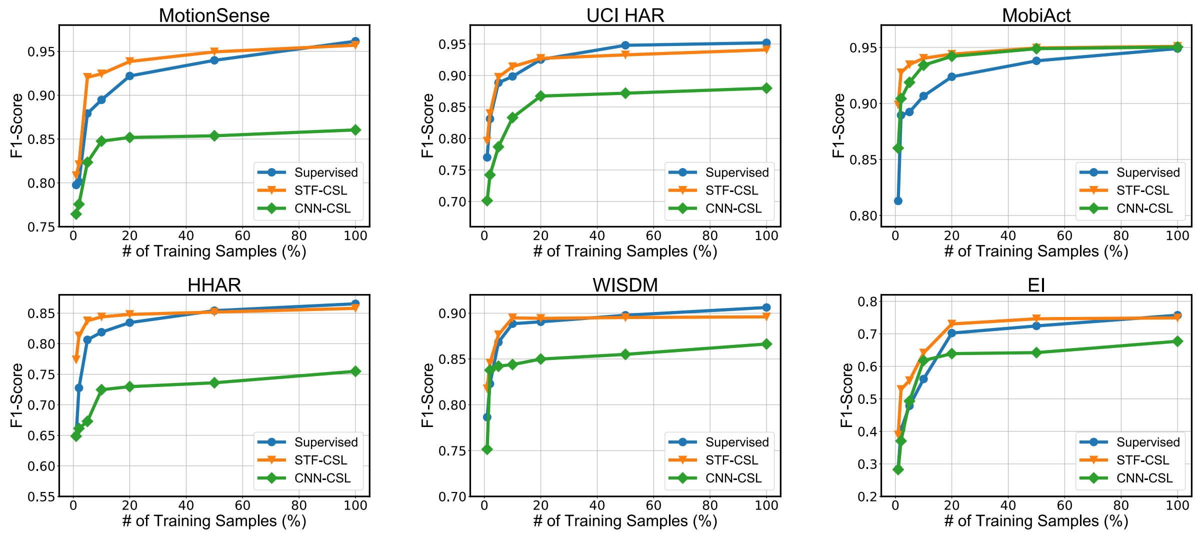

Figure 3.8: Comparison between CNN and STFNet.

means that STFNets could extract the intrinsic information from the inputs in a better way that CNNs. To illustrate the effectiveness of the contrastive self-supervised learning framework, we also compare STF-CSL with a supervised model. In the supervised model, the inputs first pass through the STFNet-bases encoder, and then the linear classifier. We train the whole model using the given labeled training data. The F1 scores of the supervised model are shown as blue curves in Figure 3.8. We observed that STF-CSL performs better than the supervised model when only a small number of labeled data items is given. This is because self-supervised pre-training has already extracted the key information of the input data. When there're enough labeled training data items, STF-CSL performs on par with the supervised model. STF-CSL usually performs not as well as the supervised model when 100% of the labels are given. Under this scenario, the performance of STF-CSL can be further improved by fine-tuning the encoder with the labeled data. We do not cover this part in our evaluation since we mainly focus on illustrating the advantages of STFNet over CNN.

Performance under Different Data Augmentation Operations Data augmentation is an important part of the contrastive self-supervised learning framework. Different data augmentation operations would lead to very different performances. In this part, we study the performance of STF-CSL under different data augmentation operations, including both time-domain and frequency-domain data augmentations, on MontionSense, MobiAct, and UCI HAR dataset. We perform the contrastive self-supervised pre-training with different data augmentation operations mentioned in section 3.4.3 and then train the linear classifier

Table 3.2: Performance under different data augmentation

Data Augmentation	MotionSense	MobiAct	UCIHAR
Jitter	0.8448	0.8455	0.7084
Rotation	0.9485	0.9410	0.9367
Permutation	0.8885	0.8848	0.7490
Time-warping	0.8685	0.8661	0.7770
Scaling	0.8726	0.8612	0.8717
Inversion	0.9159	0.9015	0.8062
Horizontally flipping	0.8946	0.8281	0.8493
Channel-shuffling	0.8421	0.9020	0.7684
Low-pass filter	0.8854	0.8441	0.8773
Amp. and phase perturbation	0.9397	0.9441	0.9396
Phase shift	0.8360	0.8578	0.8898

with 100% of labels. The results are shown in Table 3.2. We found that the performance of STF-CSL varies a lot under different data augmentation operations. For example, in MobiAct dataset, rotation performs the best among time-domain data augmentations, leading to a 0.9410 F1-score. For frequency-domain data augmentations, amplitude and phase perturbation performs the best, leading to a 0.9441 F1-score. An interesting observation is that perturbation on amplitude and phase values on the frequency-domain performs much better than directly adding Gaussian noise on time-domain values. This is because perturbing the amplitude and phase values could generate more diverse data samples with the same label, which would improve the scalability of the encoder.

Visualization of the Learned Representation We visualize the representations learned by STF-CSL through t-Distributed Stochastic Neighbor Embedding (t-SNE) [96]. t-SNE is statistical technique for visualizing high-dimensional data. It embeds high-dimensional data to a low-dimensional space of two or three dimensions by minimizing the KL divergence between them. We run the self-supervised pre-training of STF-CSL on the six HAR datasets using only unlabeled data and embed the 256-dimension output of the encoder (the activation from the global average pooling layer) to a 2D space. Then we dye each node with different colors according to their real label. The results are shown in Figure 3.9. We observe that samples with the same label could group together. This means that the contrastive self-supervised pre-training could extract the intrinsic information from the signal input without the help of labels.

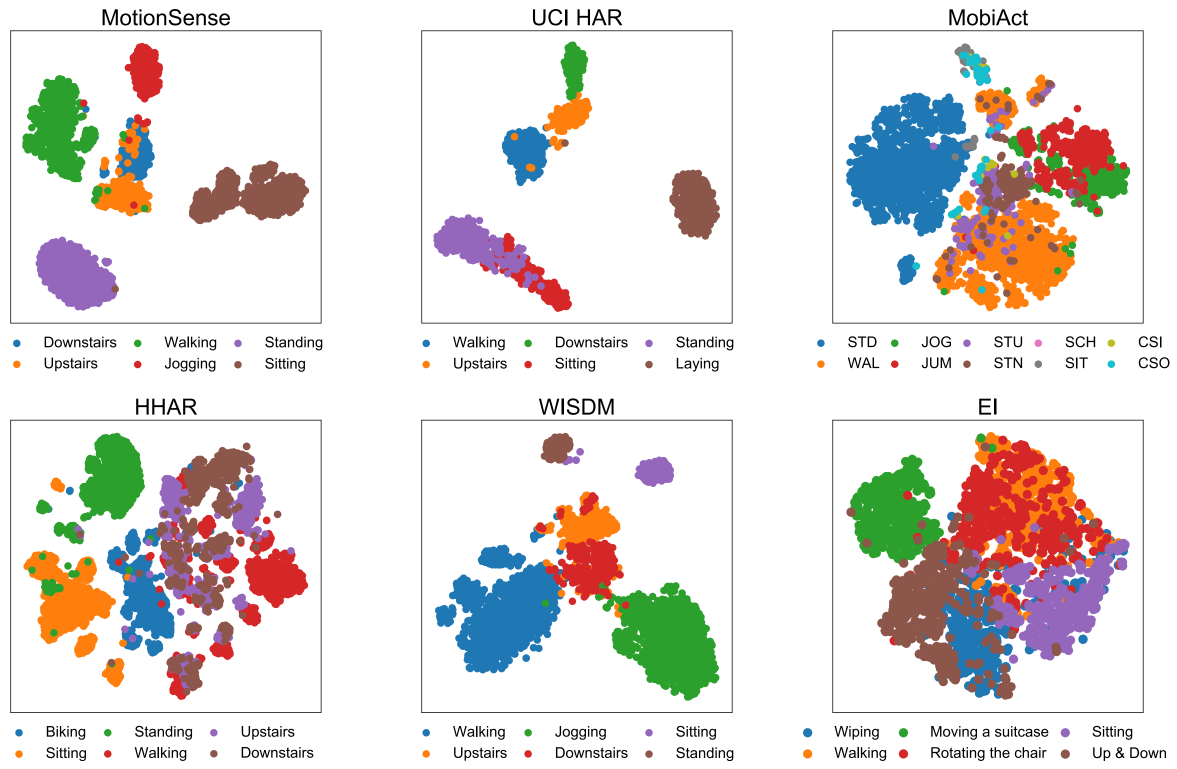

Figure 3.9: Visualization of the learned representations.

3.6 DISCUSSION AND LIMITATIONS

The work presented in this chapter shows promise in using frequency domain knowledge for contrastive learning. Improvements of up to 10% in classification accuracy are shown when we build the contrastive learning framework from frequency domain compared to contrastive learning solutions with encoders and data augmentation in the time-domain only. However, the work is far from being a comprehensive frequency domain contrastive learning framework. As such, it invites future work in multiple avenues as presented below.

Application domain limitations: The results presented in this chapter are computed for activity recognition datasets. While many datasets are used, a significant similarity exists in the time-scales of explored activities. This similarity makes it easier to tune data augmentation to the needs of this application category. It is likely that the augmentation solutions described in this work will not work out-of-the-box for other categories of IoT applications. For example, a significantly broader application domain might be one that uses RF-sensing to measure activities at different timescales from breathing and heartbeats (*e.g.*, with mm-wave radars) to various instances of vibrometry (using reflected WiFi, LoRa, ultrasonic, LiDAR, 5G, or other signals). This remains a rich avenue for future exploration.

Frequency-domain perturbation design: The frequency-domain perturbations described in this work are far from generalizable across different IoT application categories. For example, the phase shift perturbation operator essentially means that downstream AI tasks cannot use phase as input since this the data augmentation will teach the encoder to put signals of different phase shifts into the same latent cluster. For applications where phase shift measurements are important (which indeed is the case in a large fraction of RF-sensing applications), the phase shift operator should not be used for data perturbation. Alternative perturbations must be found. Similarly, the assumption that low-pass filters preserve the gist of the input signal are only true when detecting activities that evolve much slower compared to the sampling frequency used. For several categories of IoT applications, this assumption is not true. For example, in compressive sensing, measurements occur at a much lower frequency compared to the Nyquist frequency of the measured phenomenon. A low-pass filter will likely substantially degrade signal representation. A different form of filter needs to be designed for data augmentation purposes.

Time-domain perturbation design: The work inherits time-domain perturbations from prior work. There is room for significant customization of time-domain perturbations. For example, rotation and permutation operators do not really correspond to valid physical transformations on the data. Changing sensor locations does not actually result in geometric signal rotation. Also, strong auto-correlations in sensor data make arbitrary permutations invalid. For another example, reversing time by horizontal flipping often does not have a meaningful physical interpretation. Humans will not walk backwards and shattered objects will not self-reassemble. Understanding proper time-domain perturbations for IoT applications remains an open problem.

Limitations of contrastive learning: Not all local data perturbations are equally "similar" to the unperturbed signal. Understanding proper notions of similarity is very important to the design of better cost functions to use for contrastive loss in the contrastive learning framework. This work did not investigate different options for design of contrastive loss functions.

STFNet limitations: The contrastive learning framework used in this work leverages STFNet for encoder design. Different from the original STFNet approach, which works is supervised, without data augmentation, our approach combines it with the self-supervised contrastive learning framework and data augmentation. STFnet uses the Short Time Fourier Transform. Signal processing literature offers many other transforms, such as Laplace Transform and Wavelet Transform. It is interesting to investigate which transform offers better results.

Transfer learning: Self-supervised learning can be interpreted as a form of transfer learning. We can run the contrastive learning framework on a large domain-agnostic measurement dataset to train the encoder, and then reuse the parameters of the encoder as the starting point when training the downstream tasks with domain-specific labels. We leave this part as future work.

The above discussion suggests that this work is more of a conversation starter. Initial evidence suggests (using a limited study on human activity recognition applications) that frequency domain data augmentation is helpful. Perhaps the impressive part is that, despite all of the above limitations, we show accuracy gains of up to 10%. How much can be gained further if the above items are addressed? The work suggests that frequency domain data augmentation is a fertile area for future work, but such future work is beyond the scope of this chapter.

CHAPTER 4: SEMI-SUPERVISED CONTRASTIVE LEARNING FOR IOT

4.1 OVERVIEW

In this chapter, we introduce another way we apply the customization to the self-supervised contrastive learning framework for IoT applications. Specifically, we train the encoder during the self-supervised contrastive pre-training using both labeled and unlabeled data instead of only using unlabeled data as previous work did. We designed a semi-supervised (instead of self-supervised) contrastive learning framework and evaluated its performance on Human Activity Recognition (HAR) application. The evaluation shows that our approach, namely SemiC-HAR [29], demonstrates better performance compared with the original self-supervised contrastive learning framework.

The increasing number of smart phones and wearable devices yield large volumes of sensory data which can be utilized for multiple applications, including behavior and activity recognition [4, 5, 6], health and well-being [97, 98, 99], and localization and tracking [100, 101]. Specially, *Human Activity Recognition (HAR)* has been a popular application since the introduction of motion sensors (such as accelerometers and gyroscopes) on smart phones and wearable devices, offering an opportunity for understanding physical activities. Human activity recognition aims at recognizing human physical activities, such as walking, running, sitting, going downstairs, and going upstairs, based on the time-series sensory data collected from the accelerometers and gyroscopes. Traditional approaches classify human activities based on manually extracted features from the time-series data. Recent developments in deep learning demonstrated outstanding performance on various machine learning tasks [19, 20, 23], thereby inspiring the application of deep neural networks for recognizing human activities [102, 103, 104]. Novel learning paradigm emerged, such as STFNet [37] that learns sensing signals in the frequency domain, proving to be more effective compared with the traditional time-domain approaches.

Training a deep neural network to accurately recognize human activities requires a large amount of labeled data. However, it is difficult to collect labeled data for human activity recognition because the collection of the time-series sensory data and the corresponding manual annotation with activity labels must happen at the same time. This makes the HAR datasets usually laboratory-based. Although it needs a lot of human labor to label a HAR dataset, collecting raw time-series measurements from motion sensors is much easier. Hence, in order to reduce the labeling burden and effectively take advantage of the massive unlabelled sensory data, semi-supervised deep learning frameworks [4, 5, 6, 105] for HAR have been widely studied in recent years.

The semi-supervised framework for HAR is a straightforward application of general self-supervised learning techniques [106]. Self-supervised learning is a kind of unsupervised learning technique that requires mostly unlabeled data. It aims at learning representations or features that extract intrinsic structure (*e.g.*, clusters) from the original inputs with no need for labels. A downstream stage can then learn to map elements of that intrinsic structure to output categories with minimal supervision. Self-supervised contrastive learning [34] is a representative self-supervised learning framework that has gotten wide use in multiple areas due to its outstanding performance and simplicity of implementation. Self-supervised contrastive learning could help HAR applications to effectively utilize unlabeled sensory data [5]. This framework first performs self-supervised contrastive pre-training to extract intrinsic structure from the original sensory inputs, and then trains a simple (usually linear) classifier on the learned features. The contrastive pre-training runs in a self-supervised manner, where only unlabeled sensory data are used. A small amount of labeled data are used to train the classifier. Previous research has shown that labeled data can help improve the quality of features learned by contrastive learning [38]. Hence, we can get better performance if we carefully utilize both labeled and unlabeled data during the contrastive pre-training process.

To leverage this insight, we propose *SemiC-HAR*, a semi-supervised contrastive learning framework for HAR. SemiC-HAR takes both labeled and unlabeled data into consideration. The key challenge lies in how to efficiently utilize both labeled and unlabeled data in building the contrastive pre-training framework. To extract features from original inputs, previous research has studied self-supervised contrastive learning [34] which uses only unlabeled data and supervised contrastive learning [38] which uses only labeled data. In SemiC-HAR, we propose a semi-supervised contrastive learning framework to efficiently utilize both labeled and unlabeled data in the pre-training process by carefully re-designing the contrastive loss function. As far as we know, we are the first to design the semi-supervised contrastive learning framework and apply it to HAR.

In SemiC-HAR, we first train the parameters of an encoder that extracts intrinsic structure from sensory inputs to produce low-dimensional features, using the proposed semi-supervised contrastive learning framework. In this process, the exploitation of both labeled and unlabeled data and our design of the contrastive loss lead to a clear performance gain, compared with purely supervised or self-supervised approaches. After the semi-supervised contrastive pre-training, we build a linear classifier on the learned features to implement HAR. The labeled data are used to train the classifier.

The contribution of this work can be thus summarized as follows:

- We propose SemiC-HAR, a semi-supervised contrastive learning framework for HAR. SemiC-HAR first extracts features from original sensory inputs in a semi-supervised way, and then trains a classifier to recognize human activities with the labeled data. As far as we know, SemiC-HAR is the first application of a semi-supervised contrastive learning framework for HAR.

- We evaluate the performance of SemiC-HAR on six different HAR datasets, featuring different numbers of participants, devices (including different smartphones and wearable devices), and activities. We demonstrate that our semi-supervised contrastive learning framework outperforms supervised and self-supervised contrastive learning frameworks in learning features for sensory data when limited amounts of labeled data are available.

- We compared the performance of SemiC-HAR with previous supervised and semi-supervised frameworks for HAR when different amounts of labeled data are given. The evaluation results show that SemiC-HAR offers the best trade-off, compared with the previous approaches, between the amount of labeled data and the quality of activity recognition.

The rest of this chapter is organized as follows. We introduce some preliminary background and motivation in Section 4.2. Section 4.3 covers the technical details of SemiC-HAR. We describe our evaluation in Section 4.4. We finally summarize and conclude this chapter in Section 4.5.

4.2 PRELIMINARY AND MOTIVATION

In this section, we first give a preliminary overview of self-supervised contrastive learning and supervised contrastive learning, and then describe the motivation why we propose our semi-supervised contrastive learning framework for HAR.

4.2.1 Self-Supervised Contrastive Learning

Self-supervised contrastive learning is an unsupervised learning framework that learns representations or features from the inputs by maximizing agreement between differently augmented views of the same data sample via a contrastive loss in the latent space. SimCLR [34] is the representative and most commonly used self-supervised contrastive learning framework that has gotten outstanding performance in different areas like computer vision,

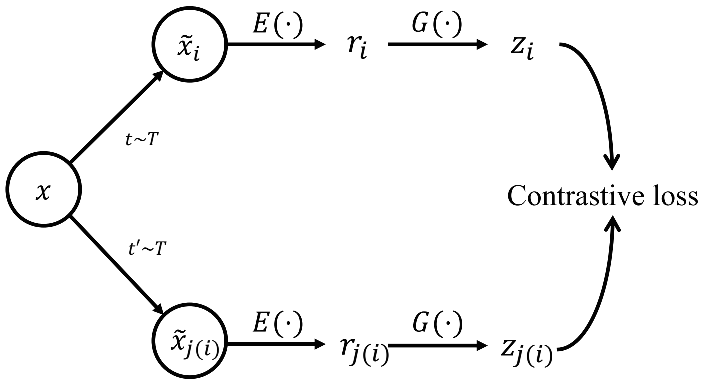

Figure 4.1: The framework of self-supervised contrastive learning (SimCLR).

automatic speech recognition, as well as IoT applications [5, 34, 107, 108]. The framework of SimCLR is shown in Figure 4.1. SimCLR mainly consists of four components.

The first part is a stochastic data augmentation module, where different views of the same data sample are generated from the same family of data augmentation operations. As shown in Figure 4.1, $\boldsymbol{x}$ refers to the original input. Two views of $\boldsymbol{x}$, here are $\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{x}}_{j(i)}$, are generated through two separate operations (t and t') from the same data augmentation family. The data augmentation operations are highly application dependent. For example, in image classification applications, random cropping, random color distortion, and Gaussian blur are commonly used data augmentation operations; in IoT applications, instead, time-warping, jitter, and channel-shuffling are popular augmentation operations when dealing with sensory data.

The second part is an encoder, E, that extracts intrinsic structure from the augmented views, leading to appropriate features or representations. Here $\boldsymbol{r}_i$ and $\boldsymbol{r}_{j(i)}$ refer to the representations of $\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{x}}_{j(i)}$:

$$\boldsymbol{r}_i = E(\tilde{\boldsymbol{x}}_i),\ \boldsymbol{r}_{j(i)} = E(\tilde{\boldsymbol{x}}_{j(i)}). \tag{4.1}$$

The third part is a projection head, G, that maps the representations to a latent space, where the contrastive loss is calculated. A common choice for the projection head is multilayer perceptron (MLP) with one or two hidden layers. For example, SimCLR applies a MLP with one hidden layer as the projection head, which means

$$\boldsymbol{z}_i = G(\boldsymbol{r}_i) = W^{(2)}\sigma(W^{(1)}\boldsymbol{r}_i), \tag{4.2}$$

where σ refers to the ReLU activation function. It has been proved in [34] that calculating the contrastive loss on $\boldsymbol{z}_i$'s instead of on the representations $\boldsymbol{r}_i$'s directly could improve the performance of contrastive learning. Hence most of the contrastive learning framework would need the projection head, G.

The last part is the contrastive loss function. The contrastive loss is defined for the contrastive prediction task which aims at maximizing the agreement between positive pairs of examples ($\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{x}}_{j(i)}$). Specifically, given a mini-batch of N data examples, the data augmentation would generate $2N$ data points. The data points generated from the same example are called a pair of positive examples. Let $i \in I \equiv \{1, 2, \ldots, 2N\}$ be the index of an arbitrary augmented data sample, and let $j(i)$ be the index of the corresponding positive examples of i. The remaining $2(N-1)$ augmented data points in the same mini-batch are called negative examples. The aim of the contrastive task is to discriminate positive pairs of examples from negative ones. The loss function for each augmented data sample i is defined as

$$\ell_i^{self} = -\log \frac{\exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_{j(i)})/\tau)}{\sum\limits_{k \in K(i)} \exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}, \tag{4.3}$$

where τ refers to the temperature parameter that controls the strength of penalties on hard negative samples and $K(i) \equiv I \setminus \{i\}$. $\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_{j(i)})$ defines the similarity between $\boldsymbol{z}_i$ and $\boldsymbol{z}_{j(i)}$. A common choice is the cosine similarity:

$$\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_{j(i)}) = \frac{\boldsymbol{z}_i^T \boldsymbol{z}_{j(i)}}{\|\boldsymbol{z}_i\| \|\boldsymbol{z}_{j(i)}\|}, \tag{4.4}$$

where $\|\boldsymbol{z}_i\|$ refers to the ℓ_2 norm of $\boldsymbol{z}_i$. The final loss function is defined as the sum of the contrastive loss across all positive pairs in a mini-batch:

$$\mathcal{L}^{self} = \sum_{i \in I} \ell_i^{self}. \tag{4.5}$$

The parameters of the encoder can be trained by minimizing the contrastive loss function in a self-supervised training process using unlabeled data. In this way, the encoder could extract the intrinsic structure from the original inputs and store it in the corresponding representations. After the self-supervised training, the parameters of the encoder are frozen. In the downstream tasks, the original inputs would pass through the encoder and a classification or regression model would build on the computed structured representations instead of the original inputs.

4.2.2 Supervised Contrastive Learning

The idea of self-supervised contrastive learning is pulling together an anchor and a single positive example (an augmented version of the same input sample) in the latent space and pushing apart the anchor from a set of negative examples consisting of the entire remainder of the mini-batch. However, among such negative examples, there may exist data samples which are in the same class as the anchor. Those samples from the same class should be considered as positive examples instead of negative ones. The self-supervised contrastive learning frameworks takes all of them as negatives because there's no labeling information and we do not know whether a data sample is from the same class with the anchor or not. If labeling information is given, the contrastive loss can be re-designed in a better way by carefully selecting the positive and negative examples for the anchor. This inspires a supervised contrastive learning framework.

Prannay et.al. propose a supervised learning framework that builds on the self-supervised contrastive learning literature by leveraging label information [38]. The contrastive loss in their supervised learning framework considers many positive examples per anchor instead of only a single positive in the self-supervised contrastive learning. Specifically, the positive examples are those in the same class with the anchor, rather than the single one which is augmented from the same input with the anchor, as done in the self-supervised contrastive learning.

Supervised contrastive learning applies the same framework with self-supervised contrastive learning, which is shown in Figure 4.1, but re-designs the contrastive loss. The contrastive loss proposed by [38], called SupCon loss, considers the scenario that more than one sample is known to belong to the same class due to the existence of labels. SupCon loss follows the following forms

$$\mathcal{L}^{sup} = \sum_{i \in I} \ell_i^{sup}, \tag{4.6}$$

where

$$\ell_i^{sup} = \frac{-1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_p)/\tau)}{\sum\limits_{k \in K(i)} \exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}. \tag{4.7}$$

Here, $P(i) \equiv \{p \in K(i) : \tilde{\boldsymbol{y}}_p = \tilde{\boldsymbol{y}}_i\}$ refers to the set of indices of all positive examples in the mini-batch which have the same label $\tilde{\boldsymbol{y}}_p$ $(= \tilde{\boldsymbol{y}}_i)$ with i. $|P(i)|$ is the cardinality of $P(i)$.

4.2.3 Motivation

In the HAR applications, we have a large amount of unlabeled sensory data and a limited amount of labeled sensory data. Early solutions used only labeled data to build supervised models for HAR. To efficiently apply unlabeled data, semi-supervised and self-supervised techniques were introduced, including contrastive learning [5]. Supervised contrastive learning has proven that labeled data could help to improve the quality of the representations learned by the contrastive learning framework. This motivates us to combine both labeled and unlabeled data in the contrastive pre-training process.

We cannot directly use the framework of supervised contrastive learning, since only a part of the data (instead all of them) is labeled. The challenge lies in defining the positive examples in a mini-batch for an anchor under a semi-supervised (instead of fully supervised) scenario. In this work, we propose a semi-supervised contrastive learning framework with a carefully designed contrastive loss to make it work in a semi-supervised scenario.

4.3 DESIGN OF SEMIC-HAR

In this section, we describe the design of our semi-supervised contrastive learning framework for HAR, namely SemiC-HAR. We first give an overview of SemiC-HAR, and then introduce the details.

4.3.1 Overview

SemiC-HAR focuses on HAR applications. Human activities can be accurately recognized through the time-series measurements of the motion sensors (*e.g.*, accelerometer and gyroscope) deployed on smartphones or wearable devices. The output of the motion sensors can be divided into segments with fixed time windows. To take advantage of unlabeled data, we consider two parts of the dataset: (i) a small labeled HAR dataset $D = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_m, \boldsymbol{y}_m)\}$ and (ii) a large unlabeled HAR dataset $U = \{\boldsymbol{x}_{m+1}, \ldots, \boldsymbol{x}_{m+n}\}$, where $\boldsymbol{x}_i$ refers to the time-series motion sensor measurements with fixed window and $\boldsymbol{y}_i$ refers to the corresponding label. The goal of our approach is to build a function $f(x_i)$, trained on both of the labeled and unlabeled data (*i.e.*, $D \cup U$), to accurately predict the activity label of each window of the time-series data $\boldsymbol{x}_i$.

Figure 4.2 illustrates the framework of SemiC-HAR. There are mainly four components:

- **Supervised Training**: We first train a "temporal classifier" C based on the labeled dataset D.

- **Self-Labeling**: The "temporal classifier" can be used to predict labels for the unlabeled dataset U. The predicted labels and the raw time-series data in U can then form

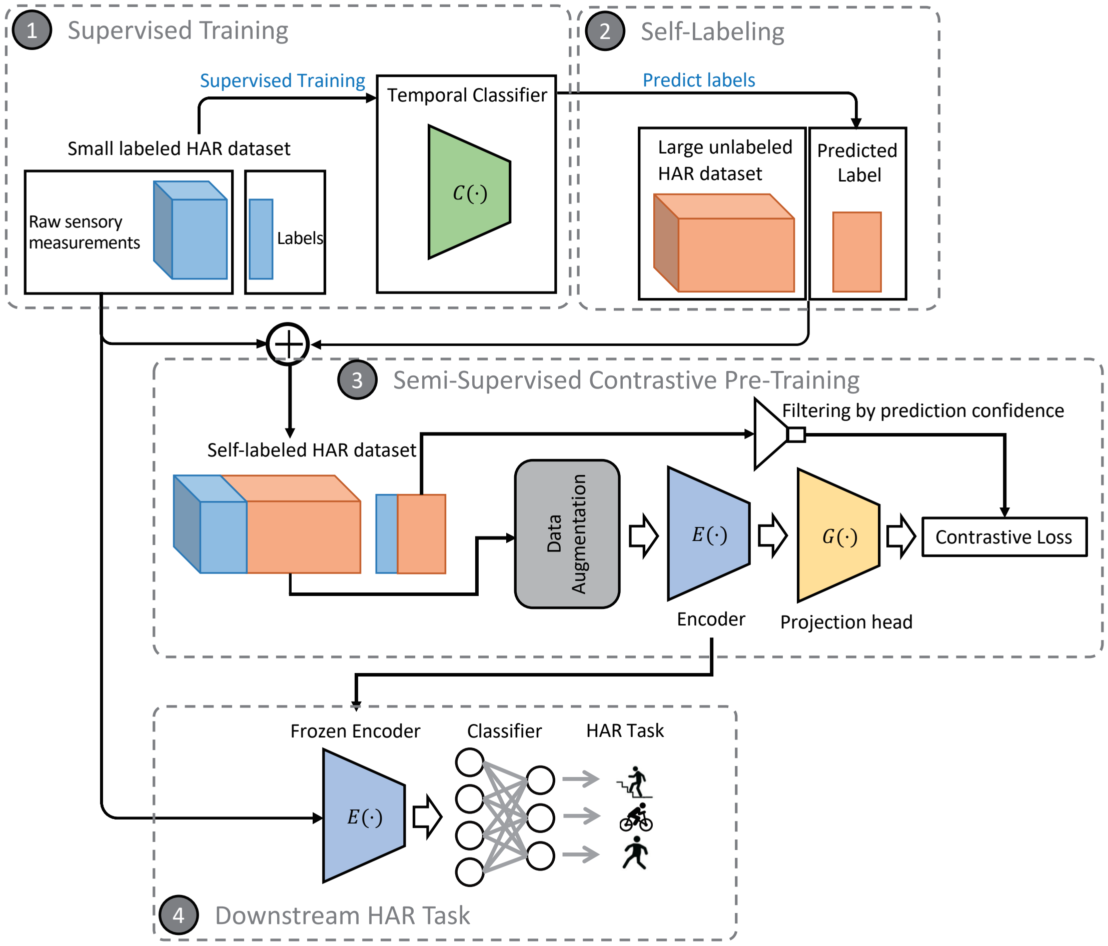

Figure 4.2: Overview of our designed semi-supervised contrastive learning framework for HAR (SemiC-HAR). SemiC-HAR consists of four components: (1) Supervised Training, (2) Self-Labeling, (3) Semi-Supervised Contrastive Learning, and (4) Downstream HAR Task.

a new labeled dataset U'. We then merge the two datasets D and U' and put them together to form the self-labeled dataset S.

- **Semi-Supervised Contrastive Pre-Training**: Based on S, we train the encoder E, that maps the raw sensory data to low-dimensional representations, through our semi-supervised contrastive learning algorithm. The encoder is then frozen and passed to the downstream HAR task.

- **Downstream HAR Task**: Given the encoder trained by the semi-supervised contrastive pre-training, we pass the time-series data from the labeled dataset D to the encoder and get the corresponding representations. Then, we train a simple classifier on the representations under a supervised way.

We will describe the details of the four components in the following subsections.

4.3.2 Supervised Training

We first train the temporal classifier C on the small labeled dataset, D. We apply STFNet [37] as the basic building block of the temporal classifier. STFNet is the Short-Time Fourier Neural Network, specially designed to deal with time-series inputs. An STFNet layer is equivalent to one layer in CNNs but it processes the input in the frequency-domain through Short-Time Fourier Transform instead of in the time-domain like CNNs. Since many human activities have cyclical motion patterns, they have clear and distinct representations in the frequency domain. STFNet has therefore shown obvious performance gains in HAR applications, compared to the traditional convolutional and recurrent neural network approach .

The temporal classifier comprises three STFNet layers, each of which followed by a dropout layer. A 1D global average pooling layer is used after the last STFNet layer. Finally, a softmax output layer is put to follow the global average pooling layer.

4.3.3 Self-Labeling

Given the temporal classifier, we label the large unlabeled dataset $U = \{\boldsymbol{x}_{m+1}, \ldots, \boldsymbol{x}_{m+n}\}$. Let $\boldsymbol{y}_i$ denote the label of $\boldsymbol{x}_i$ predicted by the temporal classifier, and $w_i \in [0,1]$ be the corresponding prediction confidence. After the self-labeling, we can get a labeled dataset $U' = \{(\boldsymbol{x}_{m+1}, \boldsymbol{y}_{m+1}, w_{m+1}), \ldots, (\boldsymbol{x}_{m+n}, \boldsymbol{y}_{m+n}, w_{m+n})\}$. For each data sample $\boldsymbol{x}_i$ in the the labeled dataset D, we also give it a prediction confidence, $w_i = 1$. We extend the dataset D in a similar way to form $D' = \{(\boldsymbol{x}_1, \boldsymbol{y}_1, w_1), \ldots, (\boldsymbol{x}_m, \boldsymbol{y}_m, w_m)\}$, where $w_i|_{i=1 \ldots m} = 1$.

We merge the two datasets to generate a new one, which we call the "Self-labeled HAR DataSet":

$$S = D' \cup U'. \tag{4.8}$$

There are raw sensor measurements, labels, and the corresponding prediction confidences in dataset S.

4.3.4 Semi-Supervised Contrastive Pre-Training

Given the self-labeled HAR dataset S, an intuitive way is to directly use the supervised contrastive learning framework we introduce in Section 4.2.2 to train the encoder E. However, the labels in S are not 100% correct because some of them are predicted by our temporal classifier. The wrong labels, that are not correctly predicted by the temporal classifier, would have side effects when designing the contrastive loss. Hence, in order to efficiently utilize the correct labels to improve the performance of contrastive learning while eliminating the side effect caused by wrong labels, we propose a modified semi-supervised contrastive learning framework.

Our approach applies the same architecture as the supervised contrastive learning (Section 4.2.2) but a different design of the contrastive loss function. Recall that in the SupCon loss function (Equation (4.6) and (4.7)) in supervised contrastive learning, all the data samples in the mini-batch that have the same label with the anchor are considered positive examples of the anchor, and the remaining data are considered negative. To eliminate the side effect caused by wrong labels, we use a filter to separate the labels into two groups: labels with high prediction confidence and labels with low prediction confidence. For those data samples with high-confidence labels, we use both of the raw sensory measurements and the labels; for those with low-confidence labels, we only use their raw sensory measurements.

Specifically, For each $\boldsymbol{x}_i$ in a mini-batch of N data examples, two views of augmented data examples, denoted by $\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{x}}_{j(i)}$, are generated from the same family of data augmentation operations. The labels and prediction confidences of $\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{x}}_{j(i)}$ are the same with $\boldsymbol{x}_i$:

$$\tilde{\boldsymbol{y}}_i = \tilde{\boldsymbol{y}}_{j(i)} = \boldsymbol{y}_i, \tag{4.9}$$

$$\tilde{w}_i = \tilde{w}_{j(i)} = w_i. \tag{4.10}$$

Let $I \equiv \{1, 2, \ldots, 2N\}$ be the index of the $2N$ augmented data examples, and $\tilde{\boldsymbol{x}}_i$ $(i \in I)$ be an arbitrary augmented data example. After $\tilde{\boldsymbol{x}}_i$ passing through the encoder E, we get the representation $\boldsymbol{r}_i = E(\tilde{\boldsymbol{x}}_i)$. The contrastive loss is calculated on the latent space of $\boldsymbol{z}_i = G(\boldsymbol{r}_i)$ after applying the projection head G. We consider two scenarios when designing our contrastive loss, for each anchor $\tilde{\boldsymbol{x}}_i$ $(i \in I)$,

- If the prediction confidence $\tilde{w}_i$ of its label $\tilde{\boldsymbol{y}}_i$ is less than or equal to a threshold TH, then $\tilde{\boldsymbol{x}}_i$ will be processed as an unlabeled data example. The contrastive loss for anchor $\tilde{\boldsymbol{x}}_i$ would be the same as the sefl-supervised contrastive learning where there is only one positive example $\tilde{\boldsymbol{x}}_{j(i)}$ which is augmented from the same data sample as $\tilde{\boldsymbol{x}}_i$.

- If the prediction confidence is larger than the threshold TH, $\tilde{\boldsymbol{x}}_i$ would be considered as a labeled data example. In this scenario, we apply a contrastive loss function which is similar but not the same with the supervised contrastive learning. We choose such data example $p \in I$ that satisfies the following two conditions as the positive examples: (i) $\tilde{\boldsymbol{x}}_p$ in the same class with $\tilde{\boldsymbol{x}}_i$ $(\tilde{\boldsymbol{y}}_p = \tilde{\boldsymbol{y}}_i)$, and (ii) the prediction confidence

$\tilde{w}_p$ is larger than the threshold TH. We also use $\tilde{w}_p$ as the weight for each component corresponding to the positive example $\tilde{\boldsymbol{x}}_p$.

Our contrastive loss can then be defined as

$$\mathcal{L}^{semi} = \sum_{i \in I} \ell_i^{semi}. \tag{4.11}$$

If $\tilde{w}_i \leq TH$,

$$\ell_i^{semi} = -\log \frac{\exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_{j(i)})/\tau)}{\sum_{k \in K(i)} \exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}. \tag{4.12}$$

If $\tilde{w}_i > TH$,

$$\ell_i^{semi} = \frac{-1}{|P'(i)|} \sum_{p \in P'(i)} \log \frac{\tilde{w}_p \exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_p)/\tau)}{\sum_{k \in K(i)} \exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}. \tag{4.13}$$

Here, $K(i) \equiv I \setminus \{i\}$; $P'(i) \equiv \{p \in K(i) : \tilde{\boldsymbol{y}}_p = \tilde{\boldsymbol{y}}_i \text{ and } \tilde{w}_p > TH\}$ refers to the set of indices of all positive examples for anchor i in the mini-batch, $|P'(i)|$ is the cardinality of $P'(i)$; and $\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)$ defines the cosine similarity between $\boldsymbol{z}_i$ and $\boldsymbol{z}_k$.

By optimizing our contrastive loss $\mathcal{L}^{semi}$, we can get the parameters of the encoder, which can be then used in the downstream HAR task. The details of our semi-supervised contrastive learning framework, including the data augmentation we used, the architectures of the encoder and projection head, are described in more detail in the evaluation Section 4.4.1.

4.3.5 Downstream HAR Task

After the optimization of the semi-supervised contrastive learning, we freeze the parameters of the encoder. In the downstream HAR task, we first map the raw measurements in the labeled dataset D to their representations, and then build a linear classifier on the representations. Since the encoder has already extracted the intrinsic information of the raw measurements, the linear classifier trained with the small labeled dataset could accurately recognize human activities. In this way, a relatively simple model trained on a small labeled dataset could accurately recognize human's activities.

4.4 EVALUATION

In this section, we introduce the experiments we have done to evaluate the performance of SemiC-HAR. We use the same datasets as mentioned in Section 3.5.1. We first describe the experiment setup including the data pre-processing and details of our approach, SemiC-HAR, and then we show the evaluation results.

4.4.1 Experiment Setup

Data Pre-Processing For EI dataset, we use all the CSI data of 30 OFDM subcarriers as the input. For the remaining 5 datasets, we apply the measurements from the accelerometers as the input. For all sensing data, we linearly interpolate them to 50Hz and segment them into sliding windows of size 512 with 50% overlap. Data from about 75% of the users in each dataset is used to training our model and the remaining is used as the test set.

Model Details The temporal classifier consists of three STFNet layers with the same kernel size 64. We set 4 resolutions $\{16, 32, 64, 128\}$ for the short-time Fourier transform of each layer. A dropout layer with a keep probability 0.8 is put after each STFNet layer. After the last STFNet layer, we use a 1D global average pooling layer followed by a softmax output layer. The encoder E has a similar structure with the temporal classifier except that E has no softmax layer and the kernel sizes of the three STFNet in E are $\{64, 64, 256\}$. This means that the representation has a length of 256. The projection head G comprises two fully connected layers with $\{256, 64\}$ hidden units. The first layer uses ReLU as its activation function and the second layer has no activation function. We set the default prediction confidence threshold to $TH = 0.9$. In the downstream HAR task, we build a linear classifier on the learned representations. We also fine-tune the last layer of the encoder when training the linear classifier. We apply a combination of rotation [4] and frequency-domain perturbation [88] as the data augmentation operation for the 5 motion sensor based datasets, and use frequency-domain perturbation when dealing with the WiFi based dataset EI.

We train our semi-supervised contrastive learning model using an Adam optimizer [95] with learning rate 10^{-4} and default decay rate. L2 regularization is added to the contrastive loss function in order to mitigate over-fitting. We train our model with a batch size 64 and 50 epochs. We have 5 runs for each experiment and calculate the average as the final performance.

4.4.2 Results

Comparison with Other HAR Frameworks We first compared SemiC-HAR with existing HAR frameworks. We consider two semi-supervised frameworks (SelfHAR[105], Transformation Discrimination[4]) and a supervised framework (STFNet[37]). Transformation Discrimination is a semi-supervised frameworks for HAR which first trains an encoder to learn the representations under a self-supervised learning model and then trains a simple classifier on the representations. Transformation Discrimination implements the self-supervision by discriminating the data examples before and after the data augmentation. SeflHAR is a semi-suerpvised HAR framework with a teacher-student model. STFNet learns sensing signals from the Time-Frequency perspective and has shown in prior work to do better for labeled time-series data. We keep 10% of the data labeled in the training set and the remaining 90% unlabeled. We use both of the labeled and unlabeled data when training semi-supervised models and use only the labeled data when training the supervised model, STFNet. Then we compare the weight-average F1-score for each of them. The results are shown in Table 4.1. We observe that SemiC-HAR performs the best compared with the previous HAR models when only 10% of the training data are labeled.

Table 4.1: Comparison with existing models

Dataset	STFNet	SelfHAR	Transformation Discrimination	SemiC-HAR
HHAR	0.8188	0.7739	0.7753	**0.8510**
MobiAct	0.9067	0.9138	0.8877	**0.9479**
MotionSense	0.8947	0.9312	0.9062	**0.9393**
UCI HAR	0.8985	0.8927	0.8915	**0.9264**
WISDM	0.8885	0.8809	0.8780	**0.9006**
EI	0.5611	0.6293	0.6049	**0.6758**

Comparison with Self-Supervised and Supervised Contrastive Learning We then compare the performance of our semi-supervised contrastive learning framework with the existing supervised and self-supervised contrastive learning frameworks when different amounts of the labeled data is given. For the self-supervised contrastive learning model, we train the encoder E based on all the raw sensory measurements in $D \cup U$. For the supervised contrasive learning model, we train the encoder based on the small labeled dataset D. The remaining parts, including the encoder, projection head, and the linear classifier, except the contrastive loss function of those three models (self-supervised, supervised, and SemiC-HAR) are the same.

The results are shown in Figure 4.3. We train the models when $5\%, 10\%, 20\%, 50\%$ and 100% of the data have labels, which means $\frac{|D|}{|D \cup U|} = 5\%, 10\%, 20\%, 50\%, 100\%$. We observe

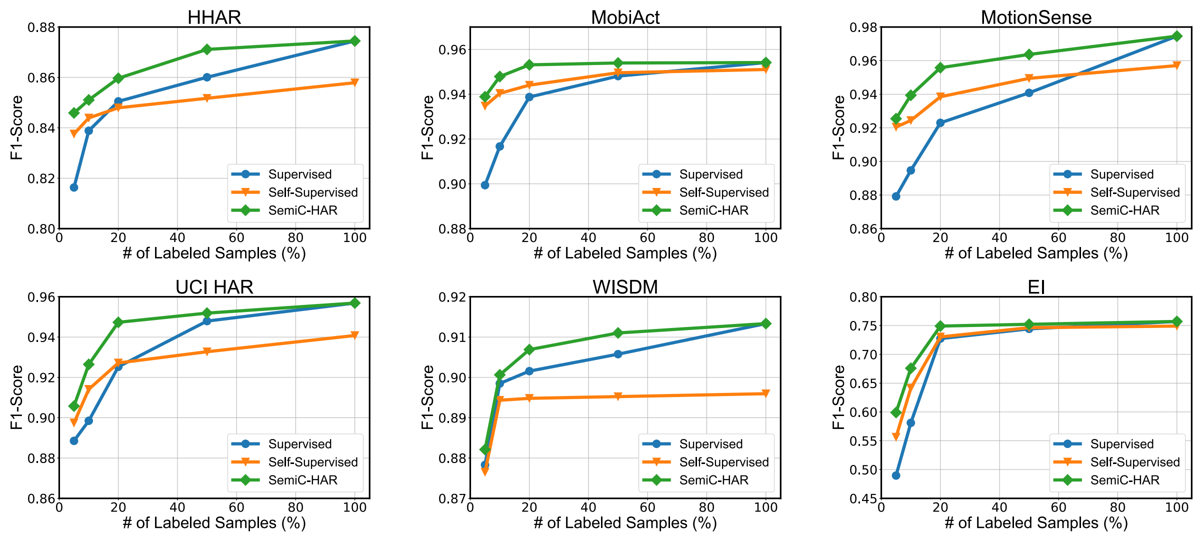

Figure 4.3: Comparison with Supervised and Self-Supervised Contrastive Learning.

that the self-supervised contrastive learning model performs better than the supervised one when the size of D is small. This is because supervised contrastive learning can only use the labeled data in D. When the size of D is larger, the supervised contrastive learning model would outperform the self-supervised model. Our semi-supervised contrastive learning framework perform better than both of them, because SemiC-HAR takes both of the labeled (D) and unlabeled (U) data when training the encoder.

Performance of Self-Labeling The performance of SemiC-HAR is highly dependent on the accuracy of the self-labeling, that is, whether the temporal classifier C trained on the small labeled dataset D could accurately annotate the raw measurements in the unlabeled dataset U. After annotating U, we can get a new labeled dataset U'. U' can be divided into two parts $U' = U'_+ \cup U'_-$ according to the prediction confidence. Here U'_+ refers to a subset of U' where the elements in U'_+ have prediction confidence larger than the threshold. U'_- refers to the subset of U' with prediction confidence less than the threshold. Since U'_+ (as well as D) provides the supervision while optimizing the contrastive loss, the size of U'_+ and the accuracy of the predicted labels in U'_+ would have impact on the performance of our approach. If most of the elements in U can be annotated with high prediction confidence (i.e., $|U'_+| \approx |U'|$) and most of the highly confident elements are correctly annotated (i.e., labels of most elements in U'_+ are correct), SemiC-HAR would perform well. We study the size of U'_+ and the accuracy of the predicted labels in U'_+ under different values of threshold (from 0.5 to 0.9) when 5%, 10%, 20%, and 50% of the training data have labels, which means $\frac{|D|}{|D \cup U|} = 5\%, 10\%, 20\%, 50\%$.

The results of "MotionSense" dataset are shown in Tabel 4.2 and Tabel 4.3. Table 4.2 shows the size of U'_+ under different thresholds when 5%, 10%, 20%, and 50% of the training data have labels. Here we use the ratio between U'_+ and U' ($|U'_+|/|U'|$) to describe the size of U'_+. We can observe that most of the elements in U can be annotated with high prediction confidence even though we use a high threshold like 0.9. Table 4.3 shows the accuracy of the predicted labels in U'_+. We observe that the temporal classifier accurately annotate the unlabeled dataset even though it is trained based on a small number of labeled dataset. The performance of the temporal classifier is good because i) STFNet performs well when limited labeled training data is given, ii) data in U and D belong to the same set of users.

A higher threshold would lead to higher accuracy but smaller size of U'_+. So there's a trade-off here. In this work, we use a fix threshold 0.9. It would be an interesting future work to study the selection of the threshold.

Table 4.2: MotionSense: Size of the high confident prediction set.

	0.5	0.6	0.7	0.8	0.9
5%	0.9998	0.9904	0.9819	0.9735	0.9645
10%	0.9995	0.9971	0.9917	0.9871	0.9804
20%	0.9988	0.9962	0.9921	0.9866	0.9782
50%	0.9995	0.9981	0.9967	0.9926	0.9879

Table 4.3: MotionSense: Accuracy of the high confident predictions

	0.5	0.6	0.7	0.8	0.9
5%	0.9456	0.9513	0.9558	0.9592	0.9652
10%	0.9757	0.9764	0.9789	0.9806	0.9834
20%	0.9913	0.9918	0.9932	0.9944	0.9951
50%	0.9935	0.9935	0.9935	0.9948	0.9953

4.5 CONCLUSION

In this chapter, we proposed SemiC-HAR, a semi-supervised contrastive learning based human activity recognition framework. We build SemiC-HAR by carefully designing the contrastive loss function which could efficiently utilize both of the labeled and unlabeled data during the contrastive pre-training. We evaluate the performance of SemiC-HAR on six different HAR datasets with different sensing signals (accelerometer, WiFi). The evaluation results demonstrate that SemiC-HAR offers an obvious performance gain compared with the existing HAR frameworks. We also compare the performance of supervised and self-supervised contrastive learning with our semi-supervised learning framework, showing the advantages of the latter.

CHAPTER 5: SPECTROGRAM MASKED AUTOENCODER FOR IOT

5.1 OVERVIEW

In this chapter, we introduce our augmentation-free self-supervised learning framework for IoT. The increasing number of smart sensing devices such as smart phones and smart wearable devices yields a large amount of sensing data which can be used to build multiple intelligent applications. Great processes have been made on studying such data, including target detection [109, 110], health and well-being [97, 98, 99], activity recognition [4, 5, 6], and localization[100, 101]. Inspired by the success gotten by the deep learning techniques on various areas like image processing [19, 20] and natural language processing (NLP) [23], a lot of deep neural network models have been designed to deal with sensing data.

As the development of deep learning techniques, deeper and deeper neural network models were designed and applied to sensing tasks. However, training such deep models requires a huge amount of labeled training data. Collecting a large dataset is not an easy job and would require a lot of human labor work. But it's observed that the workload of collecting a training dataset mainly comes from the data annotation while collecting unlabeled data is a relatively easier job. Facing this phenomenon, self-supervised learning, which is a kind of unsupervised learning technique and builds models only based on unlabeled data, has became more and more popular in dealing with sensing data.

Self-supervised learning technique was first studied in NLP and computer vision areas and has gotten great successes [23, 34, 38, 39, 111]. The goal of self-supervised learning is to learn an encoder which maps the inputs to the features or representations in some latent space. This process is only based on the unlabeled data. The encoder is then freezed and can be used in the downstream tasks like classification and regression. This means that the downstream tasks are built on the learned features instead of the original inputs. The self-supervised learning algorithms would let the features store the intrinsic information of the original input, and thus training models based on the features would need much less number of labeled training data.

Besides computer vision and NLP tasks, self-supervised learning techniques have also gotten studied in sensing tasks [29, 30, 52, 105]. Most of the previous works customized the self-supervised contrastive learning models (*e.g.*, SimCLR [34]) to sensing tasks by carefully the unique characteristics (*e.g.*, frequency domain characteristics) of sensing time-series data. The key idea of self-supervised contrastive learning is to pull together the similar samples in the latent space while pushing apart the diverse samples. Data augmentation plays an important role on the self-supervised contrastive learning technique. And it has been proved that the qualities of the features learned by the self-supervised contrastive learning largely depends on the data augmentation algorithms [5, 29]. For the same task, different data augmentation strategies would lead to very different performance. Hence, we have to carefully design the data augmentation strategy when applying self-supervised contrastive learning for a specific sensing task. What makes things worse is that different sensing tasks with different sensors usually need different data augmentation strategies. Different from computer vision and NLP tasks where the input data is image (video) or natural languages, the sensing data can be very different for different tasks. They can be measurements from accelerator, gyroscope, geophone, and microphone. Data augmentation for those data are quite different considering the different background physical phenomenon. And those data strategies that work for multiple sensing tasks, such as adding Gaussian noise, do not perform well on self-supervised contrastive learning [5]. This makes the design of self-supervised learning framework stuck in the chosen of data augmentation strategies. Hence, it would be very helpful if we could design a self-supervised learning framework for sensing tasks that does not depend on data augmentation. And that motivates this work.

The success of BERT [23] on NLP brought significant interest in the autoencoding method. And recently, [39] presented a simple, effective and scalable form of a masked autoencoder (MAE) for visual feature learning by carefully taking the difference of vision and language into consideration. Unlike the previous contrastive learning models, the MAE model does not rely on data augmentations. Hence, in this work, we built the MAE model for physical sensing data. Since the underlying processes (such as vibration or acceleration) of physical sensing are fundamentally a function of signal frequencies, the sensing data usually has a sparser and more compact representation in the frequency domain. And hence, its a good choice to use spectrograms as the input of the MAE model. The key difference between spectrogram and vision data (such as image) is that we have some prior knowledge about the spectrogram. We know that the high frequency components in the spectrogram are usually noise in the physical sensing tasks while the information mainly stores in the low frequency components. However, we do not have such information in images. In this way, the MAE should be trained by carefully reconstructing the low frequency components of the spectrograms while reconstructing the high frequency components would not help a lot. We thus re-designed the loss function of MAE by assigning different weights on different frequency components. That is, we use the weighted mean squared error (WMSE) between the original and reconstructed spectrograms as the loss function of our version of MAE. This is different from the loss function of vision based MAE which use the same weight for each pixel while calculating the mean squared error. We believe that the encoder trained with our weighted mean squared error would extract the intrinsic information from the spectrograms of sensing data in a better way, and our evaluation results confirm it.

The main contribution of this work can be summarized as follows:

- We proposed SMAE, which a Spectrogram Masked AutoEncoder framework that specially designed for physical sensing tasks. Different from the previous vision MAE, we carefully re-designed the loss function by calculating the weighted mean squared error between the original and reconstructed spectrograms. As far as we know, SMAE is the first masked autoencoder framework that designed for physical sensing applications.
- We evaluated the performance of SMAE on different datasets, where measurements of different physical sensors (such as accelerometer, gyroscope, gephpone, and microphone) are used to implement different sensing tasks like vehicle detection and human activity recognition. The experiments demonstrated the performance gain of SMAE compared with previous vision MAE. We also compared the performance of SMAE and self-supervised contrastive learning, which illustrated that SMAE does not rely on the choice of data augmentations.

The rest of this chapter is organized as follows. In Section 5.2, we introduce the background and related works. Then, we describe the detailed implementation of SMAE in Section 5.3. After that, we show the evaluation results on Section 5.4. Finally, we summarize this chapter in Section 5.5.

5.2 BACKGROUND AND RELATED WORKS

In this section, we introduce the related works. We begin with the study of deep learning on physical sensing tasks, and then have a brief introduction of the two commonly studied self-supervised learning techniques: self-supervised contrastive learning and masked autoencoder. We end this section with the self-supervised learning frameworks for sensing data.

5.2.1 Deep Learning for Physical Sensing

Inspired by the great success of deep learning techniques on computer vision and NLP [19, 20, 23, 86, 112], deep learning has gotten widely studied to deal with physical sensing data. Considering the differences between sensing data and vision/language, a lot of work has been proposed to customized deep learning techniques from vision/NLP to sensing applications by carefully taking the sensing characteristics into consideration.

Lots of illuminating works have applied deep neural network models on different sensing applications. For example, Lane et al. proposed a deep neural network based smartphone audio sensing framework, namely DeepEar [25]. Gadaleta and Rossi presented IDnet [113], which is a smartphone-based gait recognition framework built with convolutional neural networks. A smartwatch based activity recognition system was built in [114] using the Restricted Boltzmann Machines (RBM). However, those early studies did not take the temporal relationships in sensing time-series. In 2017, Yao et al. proposed DeepSense [57], which is a unified deep learning framework for time-series mobile sensing data processing. DeepSense integrates convolutional and recurrent neural networks, and exploits the interaction among different modalities of sensors and the temporal relationships which models the signal dynamics. Later, rDeepSense [115] further studied the uncertainty estimation for deep mobile sensing models.

In the study of deep learning on physical sensing, one important direction is compressing the neural network in some manner before running in the embedded device in order to meet the resource constrain of such devices. DeepIoT [58] is a commonly used compression strategy which could reduce the network size by more than 90% while keeping almost the same performance. FastDeepIoT [68] further took the non-linearity of neural network execution time into consideration and reduce the execution time without impacting accuracy. Another thing that we need to keep in mind is that the existence of enormous amount of unlabeled sensing data. Several semi-supervised learning frameworks were proposed to utilized such unlabeled sensing data [25, 116]. In this work, we design self-supervised learning framework to utilized the unlabeled data in a more general way.

5.2.2 Self-supervised Learning

Self-supervised Learning is an kind of unsupervised learning which only requires unlabeled data.The goal of self-supervised learning is to learn the representations (or features) of the input data in some low-dimensional latent space. There are mainly two stages while training a self-supervised model: pretext tasks and downstream tasks. Pretext tasks refer to the tasks for the pre-training. In this step, an encoder, which maps the input data to the representations, would be trained only based on the unlabeled data. The knowledge learned from the pretext task can be then transferred to the downstream task through the representations. Here, the downstream tasks are the target tasks which can be classification, regression, or any other machine learning tasks. And the model for the downstream task takes the representations (instead of the original input data) as input.

The supervision of self-supervised learning algorithms comes from the manually designed pretext tasks. In the early works, multiple pretext tasks, such as predicting rotation of images [31], and predicting relative position of image patches [77], have been designed to train the encoder during the pre-training process. Nowadays, there are mainly two categories of pretext tasks: contrastive learning and masked autoencoding. In the contrastive learning framework, the encoder is trained by minimizing the contrastive loss which aims at putting together the similar samples and pushing apart the diverse samples. Two representative self-supervised contrastive learning frameworks are SimCLR [34] and BYOL [117]. Both of them got great success on computer vision. One weakness of self-supervised contrastive learning is that performance of those models relies largely on the data augmentation strategies. The masked autoencoding frameworks, instead do not rely on the data augmentations. In those framework, part of the input sample is masked and the representation is learned by predicting the missing part of the input. For example, BERT [23] learns the representation for words with two pretext tasks: masked word prediction and next sentence prediction. In computer vision area, He et al. [39] showed that masked autoencoder (MAE) are scalable self-supervised learners for the image related tasks, and built a MAE model for image classification which works really well.

5.2.3 Self-supervised Learning for Sensing Data

Although self-supervised learning techniques have been widely studied in computer vision and NLP areas, the process of self-supervised leaning methods in sensing lags behind. In 2019, Saeed et al. proposed a multi-task self-supervised learning framework for human activity detection [4], which is, as far as we know, the first attempt of self-supervision for sensing representation learning. Multiple pretext tasks were designed to learn the representations. Self-supervised contrastive learning has also been applied to deal with sensing data such as measurements from accelerometer and gyroscope [5, 29, 105] and WiFi signals [30].

However, the previous models highly rely on the design of data augmentations, and the commonly used data augmentation (such as adding Gaussian noise) does not perform well. This motivates our work based on the masked autoencoder model which does not rely on the data augmentations. To the best of our knowledge, the work presented in this chapter (SMAE) is the first study of mask autoencoder on physical sensing data.

5.3 DESIGN OF SMAE

In this section, we first give an overview of our spectrogram masked autoencoder (SMAE), and then we present the detailed design including the masking, encoder, decoder, and the loss function.

5.3.1 Overview of SMAE

In SMAE, we use a same architecture as MAE [39], which is shown in Figure 5.1. Different from MAE, our model SMAE takes the spectrograms of sensing data as input. The goal of SMAE is to reconstruct the original spectrogram given its partial observation. The first step is masking, where we randomly mask part of the input spectrogram and get a partial observation. Then, the encoder would take the partial observation as input and generate the corresponding representations of the partial observation. After that, the decoder would try to take the representations as input and then reconstruct the original input spectrogam. The SMAE is trained by minimizing the difference between the reconstructed spectrogram and the original one.

SMAE adopts a similar asymmetric design with MAE, where SMAE uses a deep and complicated encoder and a lightweight decoder. After the training process, the decoder would be dropped and only the encoder would be used in the downstream tasks such as classification or regression. In the downstream tasks, the input spectrograms are first mapped to the latent representations and then neural network models are built on the representations.

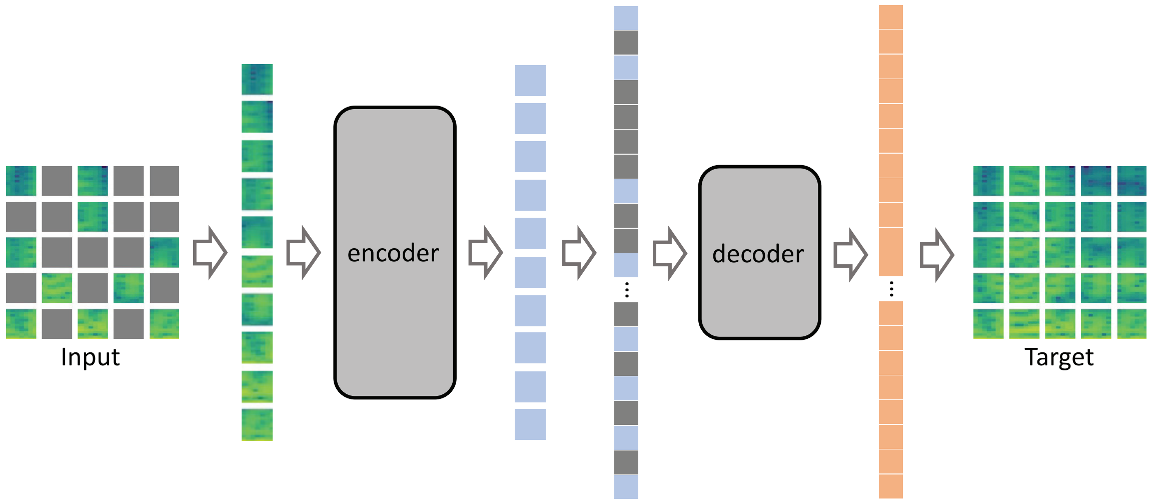

Figure 5.1: The architecture of SMAE.

5.3.2 Masking

SMAE adopts a similar masking strategy with MAE. We first divide a spectrogram into several non-overlapping patches. Then, we randomly mask a subset of the patches, shown as gray blocks in Figure 5.1. The remaining patches are kept as the input of the encoder. In order to largely eliminate the redundancy in the input spectrogram, a large masking ratio should be used (in this work 75%). In this way, the reconstruction task cannot be easily solved by simply extrapolation from the nearby unmasked patches. This makes the reconstruction task hard and thus would help the encoder to distill the intrinsic information of the input.

Different masking strategies can be used to mask the input spectrogram. In our approach, we adopt a uniform distribution to randomly mask the patches without replacement. Other mask strategies can also be adopted. For a spectrogram, different frequency bands store different information. For example, a female voice frequency range covers fairly from 350 Hz upto 17k Hz, and male voice covers a frequency range of 100 Hz to 8k Hz[1]. The frequencies of human activity are between 0 and 20 Hz [118]. This means that the information density at different frequency bands are different and the redundancies of different frequency bands are different. In this way, we may design a masking strategy that samples patches of different frequency bands with different probabilities. This kind of masking strategy could eliminate the redundancy of the spectrogram in a more fine-grained way and thus help to train an efficient encoder. In this work, we focus on the design of the loss function of SMAE and leave the design of mask strategy in the future work.

5.3.3 SMAE Encoder

In SMAE, we use vision transformer [119] as the encoder. The encoder takes the embeddings of the unmasked patches with added positional embeddings as input. The input would be passed through several layers of transformer blocks and then generates the representations. In SMAE, only a small ratio of the patches is unmasked. Even through the encoder is a very large network, it would not consume a lot of memory and computation resources.

For each unmasked patch of the input, the encoder would generate a representation vector. After training SMAE, the parameters of the encoder would be freezed and be used in the downstream tasks. In the downstream task, the encoder would take the full set of patches as input and generate representation vector for the whole input spectrogram (without masking). The target model can be built on those representation vectors.

[1]`https://seaindia.in/sea-online-journal/human-voice-frequency-range/`

5.3.4 SMAE Decoder

The encoder in SMAE generate a representation vector for each of the unmasked spectrogram patch. For each of the masked patch, we define a mask token [23], which is shared, learned vector that shows the presence of missing spectrogram patch. We concatenate the mask tokens and the representation vectors for unmasked patches together, and then add the positional embeddings of the full set to all of them. The decoder would take the sum of tokens (as well as representations) and positional embeddings as input and generate the reconstructed spectrogram.

SMAE adopts an asymmetric architecture where the designs of the encoder and decoder can be independent. In SMAE, we use vision transformer block to build the decoder, which is similar with the design of the encoder. However, the decoder is shallower and narrower than the encoder. This is because that the input size of the decoder is larger than that of the encoder. The decoder takes both of the masked tokens and unmasked representation vectors as input, but the encoder only takes the unmasked patches. The design of lightweight decoder would significantly reduce the pre-training time and computational resource consumption.

The decoder is only used in the pre-training process. After the pre-training, the decoder would be dropped, and only the encoder would be used in the downstream tasks.

5.3.5 SMAE Loss Function

The output of the SMAE decoder is a set of vectors. Each vector represents a patch in the input spectrogram. The vectors are reshape to the same size with the input spectrogram. In this way, the goal of SMAE is to predict the values of the masked (missing) patches. In our implementation, we designed weighted mean squared error (WMSE), and used it as the loss function. Here, assumes that X and $\hat{X}$ refer to the original and reconstructed spectrograms with the same size $m \times n \times c$ (c refers to the number of channels), the MAE between X and $\hat{X}$ can be computed as

$$\text{MSE} = \frac{1}{m \times n \times k} \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{c} (X_{ijk} - \hat{X}_{ijk})^2. \tag{5.1}$$

In the MAE model, only the loss on the masked part of the input is computed. Let's define the mask as a tensor M which has the same size with the spectrogram. If the elements of M are 0s, this means that the corresponding elements of the spectrogram are sampled and 1s in M means the corresponding elements in the spectrogram are masked. In this way, the

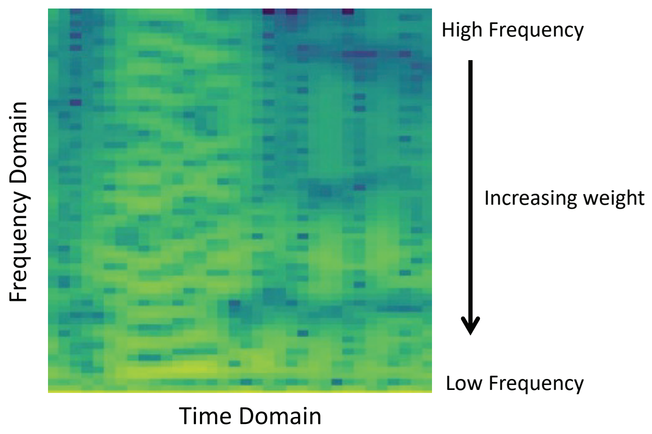

Figure 5.2: Weighted Mean Squared Error.

MAE becomes

$$\mathrm{MSE} = \frac{1}{m \times n \times k} \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{c} M_{ijk} (X_{ijk} - \hat{X}_{ijk})^2. \tag{5.2}$$

In SMAE, different from the original MAE implementation, we designed the weighted MSE (WMSE) and use it as the loss functions. This design is inspired the observation that different frequency components in a spectrogram store different information and have different importance. And the key difference between spectrogram and image is that we do have the preliminary knowledge about which part is important and which part is not. This is because of the characteristics of the physical sensing data. Minimizing the difference on the "important" components would help to train the encoder while minimizing the difference on the components that are not so important would not improve the quality of the encoder. For example, in most physical sensing tasks, such as human activity recognition and target recognition, most of the useful information locates at the low frequency parts of the spectrogram, and very high frequency parts are usually noise. Reconstructing those noise would not help a lot while training the encoder in the pre-training process. Hence, we can multiply smaller weights with those parts while calculating MSE. And for those low frequency components, we multiply them with larger weights to emphasize their importance. The WMSE thus can be defined as

$$\mathrm{WMSE} = \frac{1}{m \times n \times k} \sum_{i=1}^{m} W_i \sum_{j=1}^{n} \sum_{k=1}^{c} M_{ijk} (X_{ijk} - \hat{X}_{ijk}))^2, \tag{5.3}$$

where $i = 1, 2 \ldots, m$ refer to different frequencies in the spectrogram and W_i refers to the corresponding weights. As shown in Figure 5.2, the weight for the highest frequency is the minimum and gradually increase as the decreasing of the frequency. In detail, we set

$$W_m = W_{min}, \tag{5.4}$$

$$W_1 = W_{max}, \tag{5.5}$$

and the weights increase in a linear way, which means

$$W_i = W_{max} - \frac{(i-1)(W_{max} - W_{min})}{m-1}. \tag{5.6}$$

In the evaluation sections, we illustrate that WMSE would help to train a better encoder in the pre-training process.

5.4 EVALUATION

We show the evaluation results in this section. We study the performance of SMAE using two datasets with different sensing data. We first introduce the dataset and then describe the experiment setup. Finally, we show the evaluation results.

5.4.1 Datasets

We studied the performance of SMAE on different modalities of sensing data for two commonly studied tasks: human activities recognition and vehicle recognition. For the human activities recognition, the accelerometers and gyroscope measurements of smartphone or wearable devices can be collected and used to recognize the activities (*e.g.*, walking, sitting, running) of human beings. For the vehicle recognition task, the acoustic data (from microphone) and seismic data (from geophone) are collected and used to detect and recognize the different types of vehicles. Studying the performance of SMAE on multiple types of sensing data could demonstrate the scalability of SMAE. Now we introduce the two datasets in the evaluation.

MotionSense: MotionSense [91] is a human activity recognition dataset which consists of the measurements of accelerometer and gyroscope sensors of an iphone 6s. Different activities of 24 participants, with an iphone 6s kept in their front pocket, were recorded, and at the same time, the measurements of the two mentioned sensors were collected. To make the dataset general, the participants vary from ages, genders, heights and weights. For example, 14 males and 10 females were included in the dataset. A total of six activities were considered, including walking, jogging, sitting, standing, upstairs, and downstairs. All the accelerometer and gyroscope data was collected in 50 Hz.

ACVR: The <u>A</u>coustic and <u>S</u>eismic based <u>V</u>echicle <u>R</u>ecognition dataset (ACVR) is a dataset collected by ourselves. It's a commonly studied topic where the acoustic and seismic data can be used to do vehicle detection and classification. To collect our detaset, we build the sensor device using a Raspberry Pi connected with a microphone array and a geophone. We deployed our sensor on the grounds and collected both of the acoustic and seismic data while vehicles were driven around. We consider three types of vehicles with different engines, shapes, weights, and tires: Warthog UGV, Chevrolet Silverado, and Polaris all-terrain vehicle. The the acoustic data was collected by the microphone under 16k Hz and the seismic data was collected by the geophone under a sampling rate of 100 Hz. The data collection lasted for 115 minutes rough average split by the three types of targets.

5.4.2 Experiment Setup

In this part, we introduce the experiment setup including the data pre-processing and the implementation details of SMAE.

Data Pre-processing For MotionSense dataset, we use both of the accelerometer and gyroscope data. First, we segment the data using a sliding window of size 512 and step size of 256. 75% of the data is used as training set and the remaining 25% is used as testing.

For the acoustic and seismic dataset, considering the difference in sampling rates of the microphone and geophone, we first downsample the acoustic data to 100 Hz, which is the same as the seismic data. We performance the downsampling in the follow way: we first passed the acoustic data through a low-pass filter of 400 Hz to reduce the high frequency noise and then passed it through a high pass filter of 25 Hz in order to reduce the influence of wind. After that, we sample one point from every 160 points to get the 100 Hz acoustic data. Similar with MotionSense, we use a segment length of 512 and the ratios of training and testing sets are 75%, 25%.

We generate the spectrogram for each segment of sensing data through Short-Time Fourier Transform (STFT) with a window size 64 and step 14, and then resize the spectrogram to size 32×32. There are three dimensions for the accelerometer and gyroscope data, one dimension for the seismic data, and two channels for the acoustic data. For each dimension (or channel) of one segment sensing data, a spectrogram of size $32 \times 32 \times 2$ is generated. Here the channel size 2 refers to the real and imaginary parts of the spectrogram. Then the spectrograms of different dimensions and modalities are concatenated in the channel dimension. We then normalize the content of the spectrograms to values between 0 and 1.

Implementation Details In our SMAE model, following the architecture in Figure 5.1, we build the encoder with six layers of vision transformer and decoder with two layers. We set the length of the representation vector as 128. In each layer of the encoder and decoder, we use a same number of attention heads, which is four. In the pre-training, we use a batch size of 128. We apply AdamW [120] with a basic learning rate 10^{-3} and weight decay 10^{-4} as the optimizer. We run the pre-training a total of 200 epoches and 15% of them are used as warmup epoches.

5.4.3 Comparison with Previous Self-Supervised Approaches

We first study the performance of SMAE with previous self-supervised learning approaches on the about mentioned two datasets. To study the advantage of our weighted mean squared error, we compared the performance of SMAE with the original MAE approach which led great performance on computer vision [39]. And we also compare the performance of SMAE with the previous commonly used self-supervised contrastive learning approach SimCLR [34]. For completeness, we directly train the encoder with a linear classifier under a supervised way and compare its performance with SMAE. For the self-supervised approaches SMAE, MAE, and SimCLR, we first perform pre-training and then fine-tune the encoder as well as the linear classifier. We use the classification accuracy as the metric to study their performance.

Data augmentation is an essential part of the self-supervised contrastive learning like SimCLR. MAE and our SMAE can work without data augmentation. We can also add the data augmentation part in MAE and SMAE and the performance would get slightly improved. In this work, we use "rotation" [30] as the data augmentation algorithm for MotionSense, and "timewarp" as the data augmentation algorithm for ASVR.

The results are shown in Table 6.1. We can observe that SMAE performs the best for both of the two dataset, it shows a 95.55% accuracy for MotionSense and 95.61% accuracy for ASVR. Specially, our SMAE performs better that the supervised approach with the same structure of classifier, this means that the masked autoencoder approach can work well on dealing with sensing data. And SMAE also perform better than directly applying the original MAE approach, which designed for computer vision tasks, on sensing data (spectrogram), this means that using our weighted mean squared error would help to train a better encoder in the pre-training process.

Table 5.1: Comparison with Previous Approaches

	MotionSense	ASVR
Supervised	94.38%	93.15%
SimCLR	95.25%	93.22%
MAE	95.10%	93.95%
SMAE	**95.55%**	**94.61%**

5.4.4 Performance under Different Number of Training Data

The goal of self-supervised learning is to efficiently utilize the unlabeled data. In this part, we study the performance of SMAE when there are large amount of unlabeled data and only a small part of them has labels. We assume that only 1%, 2%, 5%, 10%, 20%, 50%, 100% of the training data has labels. The encoders of MAE, and SMAE were trained with all the training data (without label), and the models as well as the encoder in the dowstream task were fine-tuned using the labeled training data. To study the influence of unlabeled data, we also train the encoder as well as the linear classifier only based on the labeled training data and compared its performance with MAE and SMAE.

Figure 5.3 illustrates the results. We observed that both MAE and SMAE perform better than the pure supervised approach under different number of labeled training data. And the gaps between the self-supervised approaches and the supervised approach is larger when we have smaller number of labeled training data. If we compare the performance between MAE and SMAE, we found that SMAE always perform better than MAE under different number of labeled training data. This verifies the effectiveness of our designed loss function, which is weighted mean squared error between the original and reconstructed spectrograms..

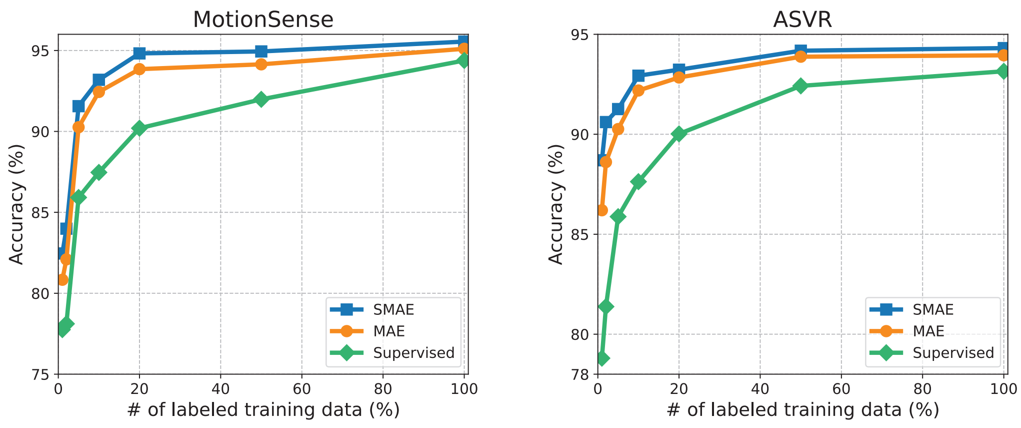

Figure 5.3: Performance under different number of labeled data.

5.4.5 Performance under Different Augmentation Strategies

One reason that we study the masked autoencdoer based self-supervised learning frameworks is that the performance of the previous commonly studied self-supervised contrastive learning approaches (*e.g.*, SimCLR) are highly dependent on the choice of data augmentation strategies. As an essential part of the contrastive learning, different data augmentations would lead to very different performances. In this part, we compare the performance between SimCLR and SMAE under different choice of data augmentation strategies. We use the MotionSense dataset in this part.

We study eight data augmentation strategies which have been widely utilized for accelerometer and gyroscope based human activity recognition [4, 5, 30].

- **Jitter**: Add random Gaussian noise to the sensing data. Jitter simulates the noise generated by the sensors.
- **Time-warp**: Time-warp operation could stretch or warp the time intervals between the sensing time-series and generate new sensing data time-series with same label as the original one.
- **Scaling:** Scaling operation generate new time-series by multiplying a random scalar to the original time-series.
- **Rotation:** The accelerometer and gyroscope are three-dimensional sensors. Rotate them should not change the labels of human activities. This data augmentation imitates the rotation fo the sensors.
- **Horizontal flipping:** Reverse the sensing data along time direction.
- **Channel-shuffling:** Randomly shuffle the channels (dimenstions) of the multi-channel sensor measurements.
- **Inversion**: Mulptiply the sensing data with -1.
- **Permutation**: First cut the sensing time-series into multiple segments along the time direction, and then permute them.

We train SimCLR and our SMAE under the above eight data augmentation strategies and compare the corresponding performance. Since data augmentation is not an essential part of SMAE, we also study the performance of SMAE without any data augmentation. We show the results in Table 5.2. We observe that the accuracies of SimCLR differ a lot under different data augmentation strategies. However, the accuracies of SMAE keep almost the same under different data augmentations. This illustrates the advantage of SMAE that it needs minimal or no data augmentation.

Table 5.2: Performance under different data augmentation

Data Augmentation	**SimCLR**	**SMAE**
No augmentation	-	94.48%
Jitter	84.81%	95.10%
Time-warp	87.22%	94.98%
Scaling	87.71%	95.30%
Rotation	95.25%	95.55%
Horizontally flipping	89.79%	95.48%
Channel-shuffling	84.70%	94.82%
Inversion	92.08%	95.40%
Permutation	87.30%	95.01%

5.5 CONCLUSION

In this chapter, we proposed SMAE, which is a masked autoencoder framework specially designed to deal with the spectrograms of physical sensing data. By carefully studying the difference between spectrogram and image, we designed weighted mean squared error (WMSE) and used it as the loss function during the pre-training process of SMAE. The weighted mean squared error allows SMAE to focus on the “important” area of the spectrograms and learn a better encoder during the pre-training. We carried out a lot of experiments to evaluate the performance of SMAE. The evaluation results verified the effectiveness of our weighted mean squared error by comparing the performance of SMAE with the original MAE model. The evaluation also demonstrated the performance gain of SMAE under different number of labels we have. Finally, the attribute that SMAE does not rely on data augmentation makes it more convenient to design self-supervised learning models for sensing data.

CHAPTER 6: A CASE STUDY: SELF-SUPERVISED LEARNING ON IOBT-OS

6.1 OVERVIEW

In this chapter, we studied the performance of our self-supervised learning framework on a real system instead of just running evaluations on different datasets. In this work, we built IOBT-OS, an operating system for the Internet of Battlefield Things where our self-supervised learning framework plays an important role on improving decision accuracy and thus optimizing decision latency. Then, we use a case study of seismic and acoustic based target detection to evaluate the performance of our proposed framework, and the evaluation result verified the effectiveness of our self-supervised learning framework.

Recent worldwide defense trends reflect an increasing investment in automating various battlefield functions [121, 122, 123]. Initial indicators suggest that related applications have already impacted the course of recent conflicts [124]. Computer communications and networks, paired with sensing and machine intelligence, are at the center of enabling technologies for these applications, making the military domain a potential key focus for the intelligent edge and Internet of Things (IoT) research. An important goal is to support performant and resilient sensor-to-decision loops at the tactical edge [125, 126]. For example, one might want to reduce latency in neutralizing threats in the battlefield. The Internet of Battlefield Things (IoBT) [127] is an operating environment for future cyber-physical battlefields, where physical and computational assets must collaborate to produce effects. Prior work articulated IoBT challenges in resilience [128, 129], distributed computing [130], Bayesian learning [131], uncertainty estimation [132] and intelligent data fusion [133, 134, 135], among other areas [136, 137, 138]. This work focuses on optimizing the efficiency and efficacy of the sensor-to-decision loop and offers an architecture, called IoBT-OS, to accomplish the optimization goals.

The need for IoBT-OS arises from the increasing complexity of networked computational resources in future battlefields. In general computing contexts, streamlining application development and operation necessitates the introduction of operating systems to address common performance and resilience challenges. Similarly, in a modern battlefield, where mission success depends in large part on computational artifacts, a new operating-system-like construct is in order. Its goal is to ensure that the execution of sensor-to-decision loops (that involve multiple intelligent devices and systems) meet the challenges arising from spatial distribution, accelerated mission-tempo, transient resources, and the potential presence of adversarial activity. IoBT provides the underlying support for ensuring performance and resilience of the networked computational and sensing substrate for the envisioned cyber-physical decision loops at the tactical edge.

The rest of this chapter is organized as follows. Section 6.2 describes a functional decomposition of the sensing-to-decision loop. Section 6.3 overviews the IoBT-OS architecture. Our experimental case-study and its evaluation are presented in Section 6.4. The chapter concludes with Section 6.5.

6.2 BACKGROUND: THE DECISION LOOP

A central concept for organizing IoBT functions is the multi-domain operation effect loop, proposed in prior work [125]. As mentioned in [125], it breaks down the data processing workflow into stages; namely, (i) detect (targets or threats), (ii) identify, (iii) track, (iv) aggregate (information), (v) distribute (to stakeholders), (vi) decide, and (vii) actuate. The execution of the loop starts with sensors that measure different signal modalities. It continues through communication components, and computational elements that execute appropriate machine analytics on sensor data. These analytics furnish information for decision making, such as detected target types and trajectories. The decisions usually involve selection of appropriate means of actuation (called *effects*) whose purpose is often to neutralize the identified threats. The success of IoBT support in executing this loop is measured against three key goals:

- *Tactical edge efficiency:* Recognizing that time is a decisive factor in gaining advantage, it is desired to accelerate the intelligent sensor-to-decision loops (ideally without loss of inference quality) and push key functionality towards the edge (to reduce dependency on large cloud-like infrastructure support). These advances allow pushing intelligent data analytics closer to the point of sensor data collection, thereby significantly shortening decision chains, while offering intelligent mission-informed data filtering as early as possible (at the edge) to reduce load on the possibly contested tactical network.

- *Edge resilience:* Recognizing that IoBT executes in adversarial settings, where new attacks on machine intelligence are possible (to disrupt decision loops and foil intelligent automation), it is important to develop foundations of resilience, especially in contexts where neural-network-based solutions are used for implementing the edge analytics. Such foundations must offer improved resilience to adversarial inputs, enhance risk analysis, speed-up sensor attack detection, and bound worst-case neural network outcomes in the presence of adversarial activity.

- *Tailored intelligence at the point of need:* Novel capabilities are needed at the tactical edge to take better advantage of heterogeneity, exploit multimodal sensing, and compute uncertainty. Examples include opportunistic exploitation of commodity radios as sensors, neural networks for inference in the frequency domain, distribution of machine learning models across heterogeneous edge systems, and sensor fusion to enhance target classification accuracy and reduce false positives (e.g., due to decoys) while meeting latency constraints.

Below, we focus primarily on edge efficiency. Issues of resilience and novel tailored intelligence services are beyond the scope of this work.

6.3 IOBT-OS

To support the above goals, the IoBT-OS architecture includes three types of modules: (i) an edge AI efficiency library, (ii) mission-informed real-time data management algorithms, (iii) digital twin support, and (iv) offline training support.

The *edge AI efficiency library* comprises modules that encapsulate different functions for target/threat detection, identification, and tracking, using neural network components that have been compressed and optimized to execute in resource-scarce environments, while offering a controllable low latency to the application.

The *mission-informed real-time data management algorithms* segment and prioritize data processing by the supported real-time edge AI components. A key challenge is to allocate more resources to the processing of more critical stimuli, which requires a combination of (i) data segmentation by some notion of data importance or urgency, and (ii) prioritized allocation of capacity to process more important/urgent data first. These algorithms should further maximize resource utilization.

Digital twin support features a set of value-added auxiliary capabilities implemented on a high-end computing platform thereby allowing the system to exploit additional resources when available (e.g., when a connection to the higher-end server can be established). A primary capability of the digital twin is to replicate key IoBT system state and environmental conditions for purposes of conducting various compute-intensive functions centrally, such as search for an optimal system configuration, global anomaly detection, root cause analysis, and model checking.

Offline training support features solutions that pre-train various models ahead of deployment. Pre-training includes execution-time profiling in order to develop accurate latency models of different system components, as well as neural network training, especially in regimes where accurate data labeling of these data is very time-consuming. A challenge is therefore to exploit mostly unlabeled data to train the neural networks involved in the sensor-to-decision loop.

IoBT-OS builds upon earlier frameworks by the authors that describe a vision of edge intelligence systems [139] and highlight challenges in applying machine intelligence to the IoT edge [140, 141].

6.4 THE CASE STUDY

6.4.1 Hardware Set-up and Execution Loop

We use a vehicle detection and classification application to study the efficiency of IoBT-OS. In our setup, a Geophone and microphone are used to collect seismic and acoustic signals for preliminary vehicle detection and classification. The sensing devices consist of a Raspberry Shake (Raspberry Pi + one vertical-axis geophone)[1], a microphone array, and a portable power bank. A camera is also used to take pictures of the targets for the final visual confirmation on a separate edge server. We used a Jetson Nano as the server.

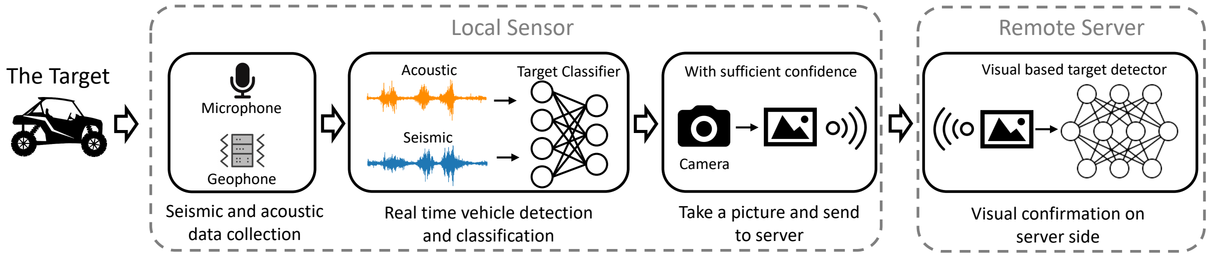
The Target
Local Sensor
Microphone
Geophone
Seismic and acoustic data collection
Acoustic
Seismic
Target Classifier
Real time vehicle detection and classification
With sufficient confidence
Camera
Take a picture and send to server
Remote Server
Visual based target detector
Visual confirmation on server side

Figure 6.1: The Execution Loop.

The execution loop is shown in Figure 6.1. The Geophone and microphone collect the seismic and acoustic signals from the environment. The seismic and acoustic data are then processed to detect and classify targets in real time. The target classifier is a DeepSense [57] neural network trained with our self-supervised contrastive learning framework. When the target classifier becomes confident enough in its prediction that a given target is found (*i.e.*, the confidence level is larger than a threshold), a message is sent to a camera that takes a picture of the target for confirmation. In order to reduce the transmission delay, the picture is first compressed by compressive offloading framework [142] and then sent to the edge server. A vision-based object detector (based on YOLO [21]) on the server processes the

[1]`https://raspberryshake.org/`

picture and confirm the target (or not). We use an object detector that was compressed using our DeepIoT [58] framework to reduce its inference delay and computational resource consumption.

6.4.2 Experimentation Results

To evaluate the performance of IoBT-OS, we deployed our devices on the grounds of the DEVCOM Army Research Laboratory Robotics Research Collaboration Campus (R2C2)[2] and collected seismic and acoustic signals, while different ground vehicles were driven around the site. Data of three different targets: a Polaris all-terrain vehicle, a Chevrolet Silverado, and Warthog UGV were collected. Each target repeatedly passed by the sensors. The total length of the experiment was 115 minutes, spread roughly equally across the three targets. The seismic data was collected at 100 Hz and the acoustic data was collected at 16k Hz. To reduce overhead, the acoustic data was low-pass filtered at 400 Hz and high-pass filtered at 25 Hz, then down-sampled to 100 Hz. Based on the collected data, we study the decision accuracy and delay of IoBT-OS. A camera was employed to simultaneously record video of the target.

Decision Accuracy We first studied the accuracy of target classification based on seismic and acoustic data. We applied the DeepSense neural network architecture [57] as the structure of the target classifier. Figure 6.2 illustrates its structure. Our previous work [37] has shown that physical sensing signals have sparser and more compact representations in the frequency domain. Thus, we used a spectrogram instead of time domain measurements as the input to the classifier. We trained the target classifier with our self-supervised contrastive learning framework.

We used roughly two thirds of the data for training and one third for testing. The seismic and acoustic data were cut into segments of one second each using a sliding window without overlap. We built a three-layers convolutional neural network (CNN) as a baseline and also compared the accuracy of our target classifier with and without self-supervised contrastive learning (SCL). The results are shown in Table 6.1. We observed that our target classifier has much higher recognition accuracy than the CNN baseline (89.1% vs 73.6%). And the that self-supervised contrastive learning further improves the accuracy from 89.1% to 91.2%.

[2]https://www.nab.usace.army.mil/Missions/Regulatory/Public-Notices/Article/492714/pn13-68-us-army-apg-g-field-and-graces-quarters-aerostat-platforms-2013-61848-m/

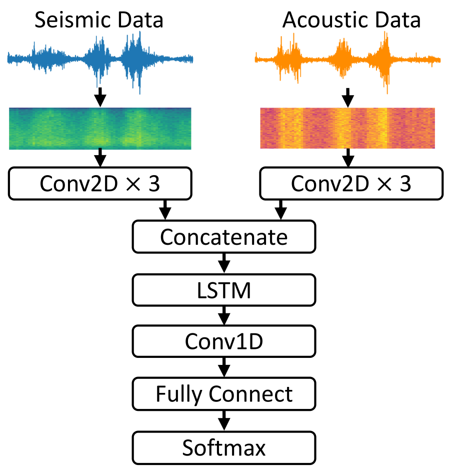

Figure 6.2: The Target Classifier.

Table 6.1: Decision Accuracy

Three-Layers CNN	**Target Classifier**	**Target Classifier+SCL**
73.6%	89.1%	91.2%

Performance under Different Data Lengths In the experiment, the seismic and acoustic signals were continuously fed into the target classifier. Given N consecutive windows (*i.e.*, N seconds window of data) we took majority vote as the final classification result. We varied N from $1s$ to $20s$ and studied the classification accuracy for different data lengths to understand the trade-off between quality and delay. The results are shown in Figure 6.3. We observed that our target classifier trained with self-supervised contrastive learning (the blue curve) always performs the best, and shows a 99% classification accuracy given 20 seconds of seismic and acoustic data.

Performance under Different Number of Labels We also studied the effectiveness of the self-supervised contrastive learning framework at utilizing unlabeled data. Specifically, we assumed that 1%, 2%, 5%, 10%, 20%, 30%, 40%, and 50% of the training data has labels (and that the remaining training data is unlabeled). We then compared the performance of the target classifier trained with supervised learning (*i.e.*, with the labeled data only) and the self-supervised contrastive learning (trained with both labeled and unlabeled data) given the above ratios of labeled data. Figure 6.4 demonstrates the results. It can be seen that the target classifier trained with our self-supervised contrstive learning (the blue curve) performs much better than that trained in a supervised manner (the orange curve). This

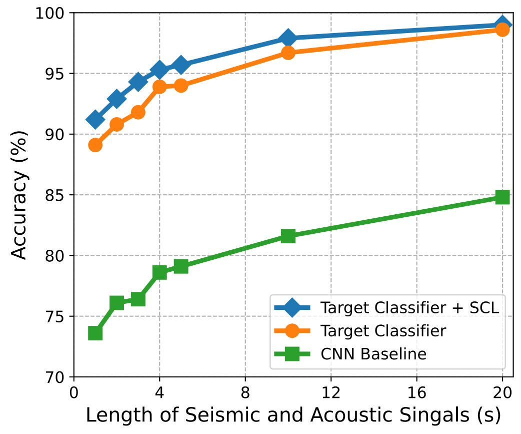

Figure 6.3: Different data lengths.

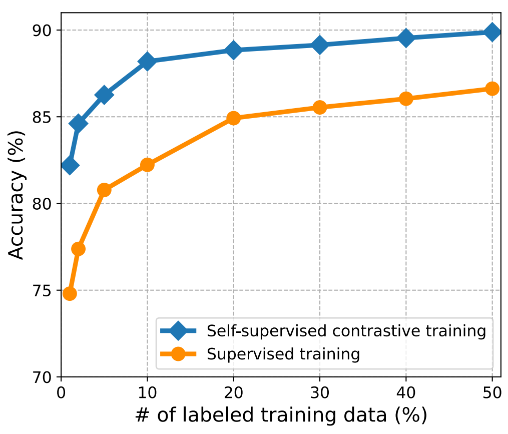

Figure 6.4: Different # of labels.

illustrates the efficiency of our self-supervised contrastive learning framework in utilizing unlabeled data.

Decision delay In this part, we study the performance of IoBT-OS on decision delay. The decision delay consists of different parts including the detection delay, the network transmission delay, as well as the delay of the virtual confirmation. Our self-supervised learning framework helps to reduce the detection delay, and thus we focus on the detection delay in this part. The detection delay comes from two parts: the time needed to get confident-enough detection results, and the inference time of the classifier. According to the results in Figure 6.3, our self-supervised learning framework as well as the DeepSense neural network reduce the time needed to get confident-enough detection results a lot compared with the CNN baseline. For the inference time of the target classifier based on acoustic and seismic sensors, the target classifier runs on the local sensors and continuously takes the seismic and acoustic data as input. It is required that the classifier's inference speed should be shorter than the interval at which seismic and acoustic data comes in. Otherwise, data will back up. Table 6.2 demonstrates the inference time (for a one second data segment as input) on different devices. We observed that the inference times of our target classifier are slightly larger than those of the three-layers CNN baseline. However, they are much smaller than one second. For example, on the Raspberry Pi 4, it takes only 17.7 ms to process the 1 second data window. This means that our classifier could generate its classification result in time.

Table 6.2: Inference Time for the Target Classifier

	Three-Layers CNN	**Target Classifier (SCL)**
Raspberry Pi 3B+	10.6 ms	66.0 ms
Raspberry Pi 4	4.7 ms	17.7 ms

6.5 CONCLUSION

In this chapter, we introduced our IoBT-OS pipeline and studied the performance of our self-supervised contrastive learning framework on a case-study of IoBT-OS. In the case study, we built a detection and confirmation loop to detect different types of vehicles based on the seismic and acoustic data collected by the low-power sensors. We collected a dataset for three different types of targets. The evaluation results demonstrated that our self-supervised learning framework can help to reduce the latency of the decision loop by improving the accuracy of the target classifier.

CHAPTER 7: SUMMARY AND FUTURE WORK

7.1 SUMMARY

In this dissertation, I studied the self-supervised learning techniques which demonstrated outstanding performance on dealing with the lack of training labels, and then customized those self-supervised learning techniques to the IoT applications. By carefully taking the unique characteristics of the IoT sensing signals into consideration, we can significantly improve the performance of self-supervised learning frameworks (originally designed for computer vision/NLP) in IoT application contexts. My customization thus provided an effective way to deal with the absence of sufficient labels in IoT applications. The work in this dissertation are summarized as follows.

Time-Domain Self-Supervised Contrastive Learning for IoT We started the customization by directly borrowing the self-supervised learning framework from computer vision and applied it to IoT applications. In this part, we applied the popular self-supervised contrastive learning framework and built a model for IoT sensing signals from the time-domain. We studied the performance of the self-supervised contrastive learning framework on the radio modulation classification dataset, and the evaluation results demonstrated the effectiveness of the self-supervised contrastive learning technique on IoT applications.

Frequency-Domain Self-Supervised Contrastive Learning for IoT We then re-designed the self-supervised contrastive learning framework by carefully studying the frequency-domain characteristics of the IoT sensing signals. We re-designed the encoder using STFNet, which is specially designed for dealing with IoT sensing signals, instead of the commonly used CNN or RNN models. Beside, we applied data augmentation from both time-domain and frequency-domain. According to our evaluation results, studying on the frequency-domain would largely improve the performance of self-supervised contrastive learning for IoT applications.

Semi-Supervised Contrastive Learning for IoT After taking the frequency-domain characteristics into consideration, we then proposed a semi-supervised contrastive learning framework for IoT. Different from the original self-supervised contrastive learning, where only unlabeled data is used in the pre-training process, our semi-supervised framework uses both of the labeled and unlabeled data to train the encoder during the pre-training process. We designed a novel way to combine the labeled and unlabeled data together, to train the encoder. Our experiments showed obvious performance gain of our semi-supervised contrastive learning framework compared with the original one.

Spectrogram Masked AutoEncoder for IoT The self-supervised contrastive learning demonstrated outstanding performance on IoT applications after our customization. However, data augmentation is an essential part of it and it requires a lot of human expertise to design a proper data augmentation strategy. Our experiments also demonstrated that the performance of the self-supervised contrastive learning framework would largely rely on the choice of data augmentation. To reduce the dependency on data augmentation, we carefully studied the masked autoencoder framework, which does not rely on the data augmentation, and customized it to IoT applications by re-designing the loss function. The evaluation results also verified the success of our customization.

A Case Study: Self-Supervised Learning on IoBT-OS We finally applied our self-supervised learning framework to a real IoT system and studied its performance on IoBT-OS. IoBT-OS is an architecture focusing on optimizing the efficiency and efficacy of the sensor-to-decision loop. We used the seismic and acoustic based vehicle recognition application as an example to evaluate our self-supervised learning framework on IoBT-OS. We carried out a lot of experiments on IoBT-OS, and they demonstrated that our self-supervised learning framework can help to reduce the sensor-to-decision latency. This shows the effectiveness of our framework on real IoT systems.

7.2 LESSONS

Data labeling is one of the key logistic costs of machine learning pipelines and this dissertation addressed the challenge of the lack of sufficient labeled data in the IoT space. By carefully customizing the self-supervised learning frameworks from computer vision/NLP to IoT applications, our self-supervised learning techniques achieved accuracy gains of up to $\times 1.5$ for IoT applications in regimes when labeled data was sparse. During the study of self-supervised learning and customizing them to IoT applications, we found some interesting and useful lessons.

Self-Supervised Contrastive Learning Self-supervised contrastive learning is an effective and popular way to utilize the unlabeled data. If we have enough expertise with application-specific data transformations that do not change data labels (*i.e.*, data augmentation), contrastive learning can get good performance. Here, a good data augmentation strategy for sensing signals is to simulate the data-distorting behavior of sensors. For the same input, we can design multiple data augmentation strategies and different augmentation strategies were proved to lead to very different performance on self-supervised contrastive learning. Hence, the top challenge in applying contrastive learning is the need of human expertise to design a good data augmentation strategy. We also found that some data augmentations are generally not helpful. For example, adding Gaussion noise is generally a bad choice because it does not exploit the underlying structure of the input data. Another challenge is that the training process of contrastive learning is easy to overfit. We should design the optimization process very carefully in oder to avoid overfitting.

Masked Autoencoding Masked autoencoding is another popular self-supervised learning architecture. Its success is based on the redundancy in the input data. If there is enough internal redundancy in the data input, masked autoencoding is generally a good choice for self-supervised learning framework. A key component of the masked autoencoding framework is the masking ratio. The mask ratio should be designed very carefully. If we use a very high mask ratio, then it would be very difficult to reconstruct the input and hence the encoder can only learn very little knowledge. If we use a very small mask ratio, then most of the input would be kept. This means that the reconstruction can be easily done by predicting the missing values according to their neighbors and the latent structure of the input could not be learned in this way. A proper mask ratio is another key point to make masked autoencoder model successful.

7.3 FUTURE WORK

In the end of this dissertation, we introduce some possible future research directions. In this dissertation, we studied the customization of self-supervised learning frameworks to IoT applications. For the future works, we cover three interesting related topics: 1) self-supervised learning frameworks for multi-modality inputs; 2) training/inferencing deep neural network models with noisy data; and 3) model compression for self-supervised learning frameworks.

Self-Supervised Learning Frameworks For Multi-Modality Inputs In this dissertation, we customized the self-supervised learning frameworks to IoT applications by taking the frequency-domain characteristic into consideration. Beside learning in frequency-domain, the IoT applications have another very unique and important characteristic, that is multi-modalities. Different from the computer vision or NLP applications, the inputs of which are images, videos, or natural language, the inputs of IoT applications are usually from multiple sensors. For example, the accelerometer and gyroscope can be collaboratively used to recognized the human activities. It would be an interesting topic to study the self-supervised learning frameworks on multi-modality inputs. There exist close correlations among the different modalities. Combining such correlations with the design of the encoder in self-supervised learning would be a valuable research topic.

Training/Inferencing Deep Neural Network Models with Noisy Data In this dissertation, we studied the self-supervised learning framework to address the challenge of the lack of sufficient labeled data in the IoT space. This is motivated by the fact that data labeling is very time-consuming and requires a lot of human labor work. It has been studied that 25% of time spent for machine learning project is allocated on data labeling[1]. And at the same time, another 25% of the time is spent on the data cleaning. Facing this observation, it would be an interesting topic to design neural network models which could deal with noisy data. Specifically, the study can be divided into two stages: 1) training neural network model with noisy data and 2) inferencing neural network models with noisy data.

Model Compression for Self-Supervised Learning Frameworks Model compression is very important for deep neural network models in IoT applications. The IoT devices usually have limited memory and computation power, and cannot support the running of large and deep neural network models. A common way to solve this problem is to compress the neural network models and then deploy the compressed models on the IoT devices. However, the previous model compression techniques performed the compression along with the supervised training process. In the future, the self-supervised learning would be more and more popular and become the major training strategy. So, studying the model compression along with the self-supervised training would be also an interesting topic.

Besides the future work mentioned above, I believe that there are a lot of other interesting topics related to deep learning on IoT. By going deeper to the deep learning techniques and designing proper customizations on IoT applications, I believe that the area of IoT would get outstanding progress in the future.

[1]`https://medium.com/whattolabel/data-labeling-ais-human-bottleneck-24bd10136e52`

REFERENCES

[1] N. Bui, A. Nguyen, P. Nguyen, H. Truong, A. Ashok, T. Dinh, R. Deterding, and T. Vu, "Pho2: Smartphone based blood oxygen level measurement systems using near-ir and red wave-guided light," in *Proceedings of the 15th ACM conference on embedded network sensor systems*, 2017, pp. 1–14.

[2] J. M. Sorber, M. Shin, R. Peterson, and D. Kotz, "Plug-n-trust: practical trusted sensing for mhealth," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*, 2012, pp. 309–322.

[3] Y. Xiang, R. Piedrahita, R. P. Dick, M. Hannigan, Q. Lv, and L. Shang, "A hybrid sensor system for indoor air quality monitoring," in *2013 IEEE International Conference on Distributed Computing in Sensor Systems.* IEEE, 2013, pp. 96–104.

[4] A. Saeed, T. Ozcelebi, and J. Lukkien, "Multi-task self-supervised learning for human activity detection," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 3, no. 2, pp. 1–30, 2019.

[5] C. I. Tang, I. Perez-Pozuelo, D. Spathis, and C. Mascolo, "Exploring contrastive learning in human activity recognition for healthcare," *arXiv preprint arXiv:2011.11542*, 2020.

[6] H. Haresamudram, A. Beedu, V. Agrawal, P. L. Grady, I. Essa, J. Hoffman, and T. Plötz, "Masked reconstruction based self-supervision for human activity recognition," in *Proceedings of the 2020 International Symposium on Wearable Computers*, 2020, pp. 45–49.

[7] E. Hoque, R. F. Dickerson, and J. A. Stankovic, "Vocal-diary: A voice command based ground truth collection system for activity recognition," in *Proceedings of the Wireless Health 2014 on National Institutes of Health*, 2014, pp. 1–6.

[8] L. Capra, W. Emmerich, and C. Mascolo, "Carisma: Context-aware reflective middleware system for mobile applications," *IEEE Transactions on software engineering*, vol. 29, no. 10, pp. 929–945, 2003.

[9] S. Nirjon, R. F. Dickerson, Q. Li, P. Asare, J. A. Stankovic, D. Hong, B. Zhang, X. Jiang, G. Shen, and F. Zhao, "Musicalheart: A hearty way of listening to music," in *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, 2012, pp. 43–56.

[10] A. Rowe, M. Berges, and R. Rajkumar, "Contactless sensing of appliance state transitions through variations in electromagnetic fields," in *Proceedings of the 2nd ACM workshop on embedded sensing systems for energy-efficiency in building*, 2010, pp. 19–24.

[11] C.-Y. Li, Y.-C. Chen, W.-J. Chen, P. Huang, and H.-h. Chu, "Sensor-embedded teeth for oral activity recognition," in *Proceedings of the 2013 international symposium on wearable computers*, 2013, pp. 41–44.

[12] E. Cho, K. Wong, O. Gnawali, M. Wicke, and L. Guibas, "Inferring mobile trajectories using a network of binary proximity sensors," in *2011 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks.* IEEE, 2011, pp. 188–196.

[13] B. Kusy, A. Ledeczi, and X. Koutsoukos, "Tracking mobile nodes using rf doppler shifts," in *Proceedings of the 5th international conference on Embedded networked sensor systems*, 2007, pp. 29–42.

[14] J. Powar, C. Gao, and R. Harle, "Assessing the impact of multi-channel ble beacons on fingerprint-based positioning," in *2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN).* IEEE, 2017, pp. 1–8.

[15] P. Lazik, N. Rajagopal, O. Shih, B. Sinopoli, and A. Rowe, "Alps: A bluetooth and ultrasound platform for mapping and localization," in *Proceedings of the 13th ACM conference on embedded networked sensor systems*, 2015, pp. 73–84.

[16] K. Langendoen and N. Reijers, "Distributed localization in wireless sensor networks: a quantitative comparison," *Computer networks*, vol. 43, no. 4, pp. 499–518, 2003.

[17] M. Mirshekari, S. Pan, P. Zhang, and H. Y. Noh, "Characterizing wave propagation to improve indoor step-level person localization using floor vibration," in *Sensors and smart structures technologies for civil, mechanical, and aerospace systems 2016*, vol. 9803. SPIE, 2016, pp. 30–40.

[18] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition.* Ieee, 2009, pp. 248–255.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[21] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[22] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[23] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[24] S. Suthaharan, "Support vector machine," in *Machine learning models and algorithms for big data classification*. Springer, 2016, pp. 207–235.

[25] N. D. Lane, P. Georgiev, and L. Qendro, "Deepear: robust smartphone audio sensing in unconstrained acoustic environments using deep learning," in *Proceedings of the 2015 ACM international joint conference on pervasive and ubiquitous computing*, 2015, pp. 283–294.

[26] T. Wang, S. Yao, S. Liu, J. Li, D. Liu, H. Shao, R. Wang, and T. Abdelzaher, "Audio keyword reconstruction from on-device motion sensor signals via neural frequency unfolding," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 5, no. 3, pp. 1–29, 2021.

[27] A. A. Hammam, M. M. Soliman, and A. E. Hassanein, "Deeppet: A pet animal tracking system in internet of things using deep neural networks," in *2018 13th International Conference on Computer Engineering and Systems (ICCES)*. IEEE, 2018, pp. 38–43.

[28] B. El Boudani, L. Kanaris, A. Kokkinis, M. Kyriacou, C. Chrysoulas, S. Stavrou, and T. Dagiuklas, "Implementing deep learning techniques in 5g iot networks for 3d indoor positioning: Delta (deep learning-based co-operative architecture)," *Sensors*, vol. 20, no. 19, p. 5495, 2020.

[29] D. Liu and T. Abdelzaher, "Semi-supervised contrastive learning for human activity recognition," in *2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, 2021, pp. 45–53.

[30] D. Liu, T. Wang, S. Liu, R. Wang, S. Yao, and T. Abdelzaher, "Contrastive self-supervised representation learning for sensing signals from the time-frequency perspective," in *2021 International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2021, pp. 1–10.

[31] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," *arXiv preprint arXiv:1803.07728*, 2018.

[32] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *European conference on computer vision*. Springer, 2016, pp. 649–666.

[33] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2536–2544.

[34] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," *arXiv preprint arXiv:2002.05709*, 2020.

[35] W. Su, X. Zhu, Y. Cao, B. Li, L. Lu, F. Wei, and J. Dai, "Vl-bert: Pre-training of generic visual-linguistic representations," *arXiv preprint arXiv:1908.08530*, 2019.

[36] T. J. O'shea and N. West, "Radio machine learning dataset generation with gnu radio," in *Proceedings of the GNU Radio Conference*, vol. 1, no. 1, 2016.

[37] S. Yao, A. Piao, W. Jiang, Y. Zhao, H. Shao, S. Liu, D. Liu, J. Li, T. Wang, S. Hu et al., "Stfnets: Learning sensing signals from the time-frequency perspective with short-time fourier neural networks," in *The World Wide Web Conference*, 2019, pp. 2192–2202.

[38] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," *arXiv preprint arXiv:2004.11362*, 2020.

[39] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," *arXiv preprint arXiv:2111.06377*, 2021.

[40] C. Long, K. Chugg, and A. Polydoros, "Further results in likelihood classification of qam signals," in *Proceedings of MILCOM'94*. IEEE, 1994, pp. 57–61.

[41] N. E. Lay and A. Polydoros, "Modulation classification of signals in unknown isi environments," in *Proceedings of MILCOM'95*, vol. 1. IEEE, 1995, pp. 170–174.

[42] O. A. Dobre, A. Abdi, Y. Bar-Ness, and W. Su, "The classification of joint analog and digital modulations," in *MILCOM 2005-2005 IEEE Military Communications Conference*. ieee, 2005, pp. 3010–3015.

[43] K. Ho, W. Prokopiw, and Y. Chan, "Modulation identification of digital signals by the wavelet transform," *IEE Proceedings-Radar, Sonar and Navigation*, vol. 147, no. 4, pp. 169–176, 2000.

[44] S. Huang, Y. Yao, Z. Wei, Z. Feng, and P. Zhang, "Automatic modulation classification of overlapped sources using multiple cumulants," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 7, 2016.

[45] T. J. OShea, J. Corgan, and T. C. Clancy, "Convolutional radio modulation recognition networks," in *International conference on engineering applications of neural networks*. Springer, 2016, pp. 213–226.

[46] T. J. OShea, T. Roy, and T. C. Clancy, "Over-the-air deep learning based radio signal classification," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 168–179, 2018.

[47] S. Rajendran, W. Meert, D. Giustiniano, V. Lenders, and S. Pollin, "Deep learning models for wireless signal classification with distributed low-cost spectrum sensors," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 3, pp. 433–445, 2018.

[48] J. Xu, C. Luo, G. Parr, and Y. Luo, "A spatiotemporal multi-channel learning framework for automatic modulation recognition," *IEEE Wireless Communications Letters*, vol. 9, no. 10, pp. 1629–1632, 2020.

[49] L. Huang, W. Pan, Y. Zhang, L. Qian, N. Gao, and Y. Wu, "Data augmentation for deep learning-based radio modulation classification," *IEEE Access*, vol. 8, pp. 1498–1506, 2019.

[50] Q. Zheng, P. Zhao, Y. Li, H. Wang, and Y. Yang, "Spectrum interference-based two-level data augmentation method in deep learning for automatic modulation classification," *Neural Computing and Applications*, pp. 1–23, 2020.

[51] P. Wang and M. Vindiola, "Data augmentation for blind signal classification," in *MILCOM 2019-2019 IEEE Military Communications Conference (MILCOM).* IEEE, 2019, pp. 305–310.

[52] D. Liu, P. Wang, T. Wang, and T. Abdelzaher, "Self-contrastive learning based semi-supervised radio modulation classification," in *MILCOM 2021-2021 IEEE Military Communications Conference (MILCOM).* IEEE, 2021, pp. 777–782.

[53] Y. Zeng, M. Zhang, F. Han, Y. Gong, and J. Zhang, "Spectrum analysis and convolutional neural network for automatic modulation recognition," *IEEE Wireless Communications Letters*, vol. 8, no. 3, pp. 929–932, 2019.

[54] E. Perenda, S. Rajendran, and S. Pollin, "Automatic modulation classification using parallel fusion of convolutional neural networks," *IEEE WIRELESS COMMUNICATIONS*, 2019.

[55] J. E. Van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Machine Learning*, vol. 109, no. 2, pp. 373–440, 2020.

[56] K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016, pp. 1857–1865.

[57] S. Yao, S. Hu, Y. Zhao, A. Zhang, and T. Abdelzaher, "Deepsense: A unified deep learning framework for time-series mobile sensing data processing," in *Proceedings of the 26th International Conference on World Wide Web*, 2017, pp. 351–360.

[58] S. Yao, Y. Zhao, A. Zhang, L. Su, and T. Abdelzaher, "Deepiot: Compressing deep neural network structures for sensing systems with a compressor-critic framework," in *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, 2017, pp. 1–14.

[59] X. Li, S. Liu, S. De Mello, X. Wang, J. Kautz, and M.-H. Yang, "Joint-task self-supervised learning for temporal correspondence," in *Advances in Neural Information Processing Systems*, 2019, pp. 318–328.

[60] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of machine learning research*, vol. 12, no. ARTICLE, pp. 2493–2537, 2011.

[61] Z. Li, Z. Xiao, B. Wang, B. Y. Zhao, and H. Zheng, "Scaling deep learning models for spectrum anomaly detection," in *Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2019, pp. 291–300.

[62] D. C. Mohr, M. Zhang, and S. M. Schueller, "Personal sensing: understanding mental health using ubiquitous sensors and machine learning," *Annual review of clinical psychology*, vol. 13, pp. 23–47, 2017.

[63] K. Han, D. Yu, and I. Tashev, "Speech emotion recognition using deep neural network and extreme learning machine," in *Fifteenth annual conference of the international speech communication association*, 2014.

[64] Y. Guan and T. Plötz, "Ensembles of deep lstm learners for activity recognition using wearables," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 2, pp. 1–28, 2017.

[65] L. Peng, L. Chen, Z. Ye, and Y. Zhang, "Aroma: A deep multi-task learning based simple and complex human activity recognition method using wearable sensors," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 2, pp. 1–16, 2018.

[66] V. Radu, C. Tong, S. Bhattacharya, N. D. Lane, C. Mascolo, M. K. Marina, and F. Kawsar, "Multimodal deep learning for activity and context recognition," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 4, pp. 1–27, 2018.

[67] S. Bhattacharya and N. D. Lane, "Sparsification and separation of deep learning layers for constrained resource inference on wearables," in *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*, 2016, pp. 176–189.

[68] S. Yao, Y. Zhao, H. Shao, S. Liu, D. Liu, L. Su, and T. Abdelzaher, "Fastdeepiot: Towards understanding and optimizing neural network execution time on mobile and embedded devices," in *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, 2018, pp. 278–291.

[69] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," in *European conference on computer vision.* Springer, 2016, pp. 69–84.

[70] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 1096–1103.

[71] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[72] G. Zhong, L.-N. Wang, X. Ling, and J. Dong, "An overview on data representation learning: From traditional feature learning to recent deep learning," *The Journal of Finance and Data Science*, vol. 2, no. 4, pp. 265–278, 2016.

[73] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[74] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[75] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[76] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[77] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1422–1430.

[78] Y. Yao, C. Liu, D. Luo, Y. Zhou, and Q. Ye, "Video playback rate perception for self-supervised spatio-temporal representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6548–6557.

[79] J. Wang, J. Jiao, L. Bao, S. He, Y. Liu, and W. Liu, "Self-supervised spatio-temporal representation learning for videos by predicting motion and appearance statistics," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4006–4015.

[80] T. Han, W. Xie, and A. Zisserman, "Self-supervised co-training for video representation learning," *arXiv preprint arXiv:2010.09709*, 2020.

[81] P. Morgado, N. Vasconcelos, T. Langlois, and O. Wang, "Self-supervised generation of spatial audio for 360 video," *arXiv preprint arXiv:1809.02587*, 2018.

[82] B. Korbar, D. Tran, and L. Torresani, "Cooperative learning of audio and video models from self-supervised synchronization," *arXiv preprint arXiv:1807.00230*, 2018.

[83] L. Kong, C. d. M. d'Autume, W. Ling, L. Yu, Z. Dai, and D. Yogatama, "A mutual information maximization perspective of language representation learning," *arXiv preprint arXiv:1910.08350*, 2019.

[84] J. O. Smith, *Mathematics of the discrete Fourier transform (DFT): with audio applications.* Julius Smith, 2007.

[85] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[86] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[87] O. Steven Eyobu and D. S. Han, "Feature representation and data augmentation for human activity classification based on wearable imu sensor data using a deep lstm neural network," *Sensors*, vol. 18, no. 9, p. 2892, 2018.

[88] J. Gao, X. Song, Q. Wen, P. Wang, L. Sun, and H. Xu, "Robusttad: Robust time series anomaly detection via decomposition and convolutional neural networks," *arXiv preprint arXiv:2002.09545*, 2020.

[89] A. Stisen, H. Blunck, S. Bhattacharya, T. S. Prentow, M. B. Kjærgaard, A. Dey, T. Sonne, and M. M. Jensen, "Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition," in *Proceedings of the 13th ACM conference on embedded networked sensor systems*, 2015, pp. 127–140.

[90] C. Chatzaki, M. Pediaditis, G. Vavoulas, and M. Tsiknakis, "Human daily activity and fall recognition using a smartphone's acceleration sensor," in *International Conference on Information and Communication Technologies for Ageing Well and e-Health.* Springer, 2016, pp. 100–118.

[91] M. Malekzadeh, R. G. Clegg, A. Cavallaro, and H. Haddadi, "Protecting sensory data against sensitive inferences," in *Proceedings of the 1st Workshop on Privacy by Design in Distributed Systems*, 2018, pp. 1–6.

[92] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones." in *Esann*, vol. 3, 2013, p. 3.

[93] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *ACM SigKDD Explorations Newsletter*, vol. 12, no. 2, pp. 74–82, 2011.

[94] W. Jiang, C. Miao, F. Ma, S. Yao, Y. Wang, Y. Yuan, H. Xue, C. Song, X. Ma, D. Koutsonikolas et al., "Towards environment independent device free human activity recognition," in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, 2018, pp. 289–304.

[95] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[96] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[97] J. S. Bauer, S. Consolvo, B. Greenstein, J. Schooler, E. Wu, N. F. Watson, and J. Kientz, "Shuteye: encouraging awareness of healthy sleep recommendations with a mobile, peripheral display," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2012, pp. 1401–1410.

[98] F. R. Bentley, Y.-Y. Chen, and C. Holz, "Reducing the stress of coordination: sharing travel time information between contacts on mobile phones," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, 2015, pp. 967–970.

[99] M. Faurholt-Jepsen, M. Vinberg, M. Frost, S. Debel, E. Margrethe Christensen, J. E. Bardram, and L. V. Kessing, "Behavioral activities collected through smartphones and the association with illness activity in bipolar disorder," *International journal of methods in psychiatric research*, vol. 25, no. 4, pp. 309–323, 2016.

[100] H. Zou, Z. Chen, H. Jiang, L. Xie, and C. Spanos, "Accurate indoor localization and tracking using mobile phone inertial sensors, wifi and ibeacon," in *2017 IEEE International Symposium on Inertial Sensors and Systems (INERTIAL)*. IEEE, 2017, pp. 1–4.

[101] E. Martin, O. Vinyals, G. Friedland, and R. Bajcsy, "Precise indoor localization using smart phones," in *Proceedings of the 18th ACM international conference on Multimedia*, 2010, pp. 787–790.

[102] N. Y. Hammerla, S. Halloran, and T. Plötz, "Deep, convolutional, and recurrent models for human activity recognition using wearables," *arXiv preprint arXiv:1604.08880*, 2016.

[103] F. J. O. Morales and D. Roggen, "Deep convolutional feature transfer across mobile activity recognition domains, sensor modalities and locations," in *Proceedings of the 2016 ACM International Symposium on Wearable Computers*, 2016, pp. 92–99.

[104] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu, "Deep learning for sensor-based activity recognition: A survey," *Pattern Recognition Letters*, vol. 119, pp. 3–11, 2019.

[105] C. I. Tang, I. Perez-Pozuelo, D. Spathis, S. Brage, N. Wareham, and C. Mascolo, "Selfhar: Improving human activity recognition through self-training with unlabeled data," *arXiv preprint arXiv:2102.06073*, 2021.

[106] I. Misra and L. v. d. Maaten, "Self-supervised learning of pretext-invariant representations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6707–6717.

[107] R. Qian, T. Meng, B. Gong, M.-H. Yang, H. Wang, S. Belongie, and Y. Cui, "Spatiotemporal contrastive video representation learning," *arXiv preprint arXiv:2008.03800*, 2020.

[108] D. Jiang, W. Li, M. Cao, R. Zhang, W. Zou, K. Han, and X. Li, "Speech simclr: Combining contrastive and reconstruction objective for self-supervised speech representation learning," *arXiv preprint arXiv:2010.13991*, 2020.

[109] G. Jin, B. Ye, Y. Wu, and F. Qu, "Vehicle classification based on seismic signatures using convolutional neural network," *IEEE Geoscience and Remote Sensing Letters*, vol. 16, no. 4, pp. 628–632, 2018.

[110] G. P. Mazarakis and J. N. Avaritsiotis, "Vehicle classification in sensor networks using time-domain signal processing and neural networks," *Microprocessors and MICROSYSTEMS*, vol. 31, no. 6, pp. 381–392, 2007.

[111] M. Kim, J. Tack, and S. J. Hwang, "Adversarial self-supervised contrastive learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 2983–2994, 2020.

[112] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[113] M. Gadaleta and M. Rossi, "Idnet: Smartphone-based gait recognition with convolutional neural networks," *Pattern Recognition*, vol. 74, pp. 25–37, 2018.

[114] S. Bhattacharya and N. D. Lane, "From smart to deep: Robust activity recognition on smartwatches using deep learning," in *2016 IEEE International conference on pervasive computing and communication workshops (PerCom Workshops)*. IEEE, 2016, pp. 1–6.

[115] S. Yao, Y. Zhao, H. Shao, A. Zhang, C. Zhang, S. Li, and T. Abdelzaher, "Rdeepsense: Reliable deep mobile computing models with uncertainty estimations," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 4, pp. 1–26, 2018.

[116] S. Yao, Y. Zhao, H. Shao, C. Zhang, A. Zhang, S. Hu, D. Liu, S. Liu, L. Su, and T. Abdelzaher, "Sensegan: Enabling deep learning for internet of things with a semi-supervised framework," *Proceedings of the ACM on interactive, mobile, wearable and ubiquitous technologies*, vol. 2, no. 3, pp. 1–21, 2018.

[117] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar et al., "Bootstrap your own latent-a new approach to self-supervised learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 271–21 284, 2020.

[118] E. K. Antonsson and R. W. Mann, "The frequency content of gait," *Journal of biomechanics*, vol. 18, no. 1, pp. 39–47, 1985.

[119] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly et al., "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[120] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[121] A. B. Assessment and C. P. Brief, "US military investments in autonomy and AI," 2020.

[122] S. Petrella, C. Miller, and B. Cooper, "Russia's artificial intelligence strategy: the role of state-owned firms," *Orbis*, vol. 65, no. 1, pp. 75–100, 2021.

[123] M. C. Horowitz, "Artificial intelligence, international competition, and the balance of power," *2018*, vol. 22, 2018.

[124] J. F. Antal, *7 Seconds to Die: A Military Analysis of the Second Nagorno-Karabakh War and the Future of Warfighting.* Casemate, 2022.

[125] T. Abdelzaher, A. Taliaferro, P. Sullivan, and S. Russell, "The multi-domain operations effect loop: from future concepts to research challenges," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications II*, vol. 11413. International Society for Optics and Photonics, 2020, p. 1141304.

[126] S. Russell, T. Abdelzaher, and N. Suri, "Multi-domain effects and the internet of battlefield things," in *MILCOM 2019-2019 IEEE Military Communications Conference (MILCOM).* IEEE, 2019, pp. 724–730.

[127] S. Russell and T. Abdelzaher, "The internet of battlefield things: the next generation of command, control, communications and intelligence (c3i) decision-making," in *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM).* IEEE, 2018, pp. 737–742.

[128] T. Abdelzaher, N. Ayanian, T. Basar, S. Diggavi, J. Diesner, D. Ganesan, R. Govindan, S. Jha, T. Lepoint, B. Marlin et al., "Toward an internet of battlefield things: A resilience perspective," *Computer*, vol. 51, no. 11, pp. 24–36, 2018.

[129] S. Liu, S. Yao, Y. Huang, D. Liu, H. Shao, Y. Zhao, J. Li, T. Wang, R. Wang, C. Yang et al., "Handling missing sensors in topology-aware iot applications with gated graph neural network," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, no. 3, pp. 1–31, 2020.

[130] T. Abdelzaher, N. Ayanian, T. Basar, S. Diggavi, J. Diesner, D. Ganesan, R. Govindan, S. Jha, T. Lepoint, B. Marlin et al., "Will distributed computing revolutionize peace? the emergence of battlefield IoT," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS).* IEEE, 2018, pp. 1129–1138.

[131] A. D. Cobb, B. A. Jalaian, N. D. Bastian, and S. Russell, "Robust decision-making in the internet of battlefield things using bayesian neural networks," in *2021 Winter Simulation Conference (WSC).* IEEE, 2021, pp. 1–12.

[132] B. M. Marlin, T. Abdelzaher, G. Ciocarlie, A. D. Cobb, M. Dennison, B. Jalaian, L. Kaplan, T. Raber, A. Raglin, P. K. Sharma et al., "On uncertainty and robustness in large-scale intelligent data fusion systems," in *2020 IEEE Second International Conference on Cognitive Machine Intelligence (CogMI)*. IEEE, 2020, pp. 82–91.

[133] E. Blasch, T. Pham, C.-Y. Chong, W. Koch, H. Leung, D. Braines, and T. Abdelzaher, "Machine learning/artificial intelligence for sensor data fusion–opportunities and challenges," *IEEE Aerospace and Electronic Systems Magazine*, vol. 36, no. 7, pp. 80–93, 2021.

[134] S. Liu, S. Yao, J. Li, D. Liu, T. Wang, H. Shao, and T. Abdelzaher, "Giobalfusion: A global attentional deep learning framework for multisensor information fusion," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, no. 1, pp. 1–27, 2020.

[135] S. Yao, Y. Zhao, H. Shao, D. Liu, S. Liu, Y. Hao, A. Piao, S. Hu, S. Lu, and T. F. Abdelzaher, "Sadeepsense: Self-attention deep learning framework for heterogeneous on-device sensors in internet of things applications," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1243–1251.

[136] J. Huang, C. Samplawski, D. Ganesan, B. Marlin, and H. Kwon, "Clio: Enabling automatic compilation of deep learning pipelines across iot and cloud," in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, 2020, pp. 1–12.

[137] T. Li, J. Huang, E. Risinger, and D. Ganesan, "Low-latency speculative inference on distributed multi-modal data streams," in *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, 2021, pp. 67–80.

[138] D. Basu, D. Data, C. Karakus, and S. Diggavi, "Qsparse-local-sgd: Distributed sgd with quantization, sparsification and local computations," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[139] S. Yao, Y. Hao, Y. Zhao, A. Piao, H. Shao, D. Liu, S. Liu, S. Hu, D. Weerakoon, K. Jayarajah et al., "Eugene: Towards deep intelligence as a service," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019, pp. 1630–1640.

[140] S. Yao, Y. Zhao, A. Zhang, S. Hu, H. Shao, C. Zhang, L. Su, and T. Abdelzaher, "Deep learning for the internet of things," *Computer*, vol. 51, no. 5, pp. 32–41, 2018.

[141] T. Abdelzaher, Y. Hao, K. Jayarajah, A. Misra, P. Skarin, S. Yao, D. Weerakoon, and K.-E. Årzén, "Five challenges in cloud-enabled intelligence and control," *ACM Transactions on Internet Technology (TOIT)*, vol. 20, no. 1, pp. 1–19, 2020.

[142] S. Yao, J. Li, D. Liu, T. Wang, S. Liu, H. Shao, and T. Abdelzaher, "Deep compressive offloading: Speeding up neural network inference by trading edge computation for network latency," in *Proceedings of the International Conference on Embedded Networked Sensor Systems (SenSys)*, 2020.