UNIVERSITY OF CALIFORNIA SAN DIEGO

Efficient Learning for Sensor Time Series in Label Scarce Applications

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Computer Science

by

Ranak Roy Chowdhury

2024

University of California San Diego

2024

TABLE OF CONTENTS

Dissertation Approval Page iii

Table of Contents iv

List of Figures vii

List of Tables viii

List of Algorithms x

Acknowledgements xi

Vita xiv

Abstract of the Dissertation xvi

Chapter 1 Introduction 1
1.1 Growth of Sensing Technologies 1
1.2 The Label Scarcity Problem 3
1.3 Self-Supervised Learning (SSL) 4
1.4 Learning Pipeline 6
1.5 Human Activity Recognition (HAR) 7
1.6 Proposed Framework 9
1.7 Contribution and Outline 10

Chapter 2 Time-interval Aware Self-supervision 13
2.1 Problem Statement 14
2.2 Related Work 17
2.2.1 Self-Supervised Regular Time Series Methods 17
2.2.2 Irregular Time Series Methods 18
2.3 Notation 18
2.4 Model Architecture 19
2.5 PrimeNet Framework 21
2.5.1 Time Contrastive Learning (TimeCL) 21
2.5.2 Time Reconstruction (TimeReco) 23
2.6 Experiments 28
2.6.1 Datasets 28
2.6.2 Baselines 30
2.6.3 Experimental Protocols 31
2.6.4 Results 33
2.6.5 Ablations 34
2.7 Conclusion 35

Chapter 3 Task-specific Self-supervision . . . 36
3.1 Problem Statement . . . 37
3.2 Related Work . . . 39
3.2.1 Non-Deep Learning Methods . . . 39
3.2.2 Deep Learning Methods . . . 40
3.3 TARNet Framework . . . 41
3.3.1 Problem Description and Notations . . . 41
3.3.2 Base Model . . . 41
3.3.3 End Task (T_{END}) . . . 42
3.3.4 Task-aware Reconstruction (T_{TAR}) . . . 43
3.3.5 Data-driven Masking Strategy (M) . . . 45
3.4 Experiments . . . 47
3.4.1 Experimental Setup . . . 48
3.4.2 Evaluation Metrics . . . 50
3.4.3 Classification . . . 51
3.4.4 Regression . . . 52
3.4.5 Ablation Study . . . 53
3.4.6 Can T_{TAR} compensate for limited labeled training data? . . . 54
3.4.7 Explaining Masking Strategy, M . . . 55
3.5 Conclusion . . . 58

Chapter 4 Sensor Context Aware Self-supervision . . . 64
4.1 Problem Statement . . . 65
4.2 Related Works . . . 69
4.3 ZeroHAR Framework . . . 71
4.3.1 Problem Statement . . . 71
4.3.2 Stage I: IMU-Text Alignment . . . 72
4.3.3 Stage II: Action Recognition . . . 76
4.4 Experimental Results . . . 79
4.4.1 Dataset, Baselines and Metrics . . . 79
4.4.2 Experimental Setup . . . 84
4.4.3 Results . . . 85
4.4.4 Ablations . . . 87
4.4.5 Case Study on IMU-Text Alignment . . . 87
4.5 Conclusion . . . 88

Chapter 5 Conclusion and Future Works . . . 90
5.1 Limitations and Open Challenges . . . 91

Appendix A Time-interval Aware Self-supervision . . . 93
A.0.1 Notations . . . 93

Appendix B Task-specific Self-supervision . . . 95
B.0.1 Notations . . . 95

B.0.2 Environment 95
B.0.3 Training settings 95

Appendix C Sensor Context Aware Self-supervision 99
C.0.1 Notations 99
C.0.2 Experimental Details 99

Bibliography 102

LIST OF FIGURES

Figure 1.1. The number of sensors covering diverse application domains, as well as their revenue has significantly increased over the years. 2

Figure 1.2. Subject follows predefined protocol. 2

Figure 1.3. Subject conducts activities in-the-wild and a human annotator watches the subject and labels the activities. 2

Figure 1.4. Pipeline of our proposed label-efficient sensor time series analysis framework. 9

Figure 2.1. Augmenting time-series through different sampling techniques. 14

Figure 2.2. Difference between constant time and constant length masking. 14

Figure 2.3. Subsampling from irregular time-series results in non-representative samples. 15

Figure 2.4. PrimeNet Model Architecture Overview. 20

Figure 2.5. PrimeNet Pre-training and Fine-tuning Overview. 27

Figure 3.1. TARNet Overview.. 37

Figure 3.2. (a) and (b) show classification accuracy, and (c) and (d) show regression RMSE Loss against % of training data. 55

Figure 3.3. σ plotted as heatmap for Epilepsy. 56

Figure 3.4. σ plotted as heatmap for Face Detection. 57

Figure 4.1. Different Approaches to Zero-shot Wearable HAR. 66

Figure 4.2. ZeroHAR Training Overview. 67

Figure 4.3. ZeroHAR Architectural and Loss Overview. 70

Figure 4.4. Training, validation, and test data splits. 84

Figure 4.5. Scaling Laws and Loss Convergence of ZeroHAR. 86

Figure 4.6. t-SNE vizualization of ZeroHAR on PAMAP2. 88

LIST OF TABLES

Table 1.1. Metric and directionality of IMU sensors. 8

Table 1.2. Summary of contributions of our proposed data-efficient learning scheme for label scarce sensory applications. 10

Table 2.1. Classification on PhysioNet [106] measured by Area Under the ROC curve (AUC). Higher AUC is better. 28

Table 2.2. Classification on MIMIC-III [58] measured by Area Under the ROC curve (AUC). Higher AUC is better. 29

Table 2.3. Interpolation on PhysioNet at 50% [106] ($\times 10^{-2}$), measured by RMSE. Lower RMSE is better. 30

Table 2.4. Interpolation on Activity at 10% [59] ($\times 10^{-2}$), measured by RMSE. Lower RMSE is better. 31

Table 2.5. Regression on Appliances Energy [115], measured by RMSE. Lower RMSE is better. 32

Table 2.6. Classification performance of PrimeNet's ablation study on PhysioNet. . . . 32

Table 3.1. Accuracy of ED, MLSTM-FCNs, DTWD, TapNet, and DTWI. 59

Table 3.2. Accuracy of NS, WEASEL-MUSE, TS-TCC, TNC, and ShapeNet. 60

Table 3.3. Accuracy of TS2Vec, Rocket, MiniRocket, TST, and TARNet. 61

Table 3.4. Root Mean Squared Error (RMSE) of 1-NN-DTWD, 1-NN-ED, 5-NN-ED, and 5-NN-DTWD. 62

Table 3.5. Root Mean Squared Error (RMSE) of SVR, ResNet, FCN, and Rocket. . . . 62

Table 3.6. Root Mean Squared Error (RMSE) of Inception, RF, XGB, TST, and TARNet. 63

Table 3.7. Ablation study of TARNet 63

Table 4.1. Datasets used. Each body position is equipped with all the sensors specified under sensors column. 80

Table 4.2. Zero-shot classification on Opportunity, PAMAP2, and Harth. 81

Table 4.3. Zero-shot classification on USCHAD and WISDM. 82

Table 4.4. Regular HAR Classification Performance. 82

Table 4.5. Ablation Study of ZeroHAR on Opportunity, PAMAP2, and Harth. 83

Table 4.6. Ablation Study of ZeroHAR on USCHAD and WISDM. 83

Table A.1. Notations used in Chapter 2. 94

Table B.1. Notations used in Chapter 3. 96

Table B.2. Best hyper-parameter settings for classification experiments. 97

Table B.3. Best hyper-parameter settings for regression experiments. 98

Table C.1. Notations used in Chapter 4. 100

Table C.2. Relative performance comparison of ZeroHAR with 2nd best results. 101

Table C.3. Class splits across 3 folds. 101

LIST OF ALGORITHMS

Algorithm 1. TimeCL Data Augmentation 22

Algorithm 2. TimeReco Data Augmentation 24

Algorithm 3. Training of TARNet 45

Algorithm 4. Stage I: IMU-Text Alignment 76

Algorithm 5. Stage II: Action Recognition 79

ACKNOWLEDGEMENTS

This dissertation would not have been possible without the support of many remarkable individuals I've had the privilege to meet.

I extend my deepest gratitude to my advisors, Rajesh K. Gupta and Jingbo Shang, for providing me the opportunity to pursue my PhD as your student and your unwavering support and trust throughout my academic journey. Rajesh, your intellectual rigor and profound thinking have significantly shaped my development as an independent researcher. Your mentorship has expanded my intellectual horizons by teaching me how to think critically and creatively. I am incredibly grateful for the academic freedom you provided, allowing me to explore my research interests while consistently guiding me back on track whenever I felt lost. Jingbo, I have gained so much from your technical acumen and attention to detail. Thank you for the countless hours of enlightening meetings, detailed and focused feedback. Your technical expertise, combined with your down-to-earth attitude and humility, has inspired me to strive for excellence while staying grounded. I consider myself incredibly lucky to have you both as my advisors. Thank you for your unwavering support, encouragement, and career guidance, and for being such compassionate mentors who care not only about our research but also about our well-being as PhD students. Under your exemplary leadership, I have found a nurturing environment to learn, grow, and thrive, for which I am profoundly thankful.

I am incredibly thankful to Sicun Gao and Tauhidur Rahman for serving on my thesis committee and providing valuable feedback to my thesis and presentations. Special thanks to Dean Tullsen and Rose Yu for joining my research exam committee.

I would like to express my special gratitude to Dezhi Hong and Xiyuan Zhang for their significant contributions to my doctoral research. Working alongside you has been a privilege as it taught me invaluable skills that have greatly contributed to my development as a researcher. It has helped me to navigate the challenges of research, and I am deeply grateful for the knowledge and insights you have shared with me. I would like to express my heartfelt gratitude to my undergraduate thesis advisor, Muhammad Abdullah Adnan, who was the first to inspire and guide me in the world of research. His mentorship and steadfast encouragement played a pivotal role in my journey to pursue a PhD, and I am deeply thankful for his influence on my academic career.

I am grateful to Shuheng Li, Xiaohan Fu, Jiayun Zhang, Hsin-Yu Liu, Jiacheng Li, Ameya Panse, Ritvik Kapila, and Nachiket Mokashi for their insightful contributions to my research projects. Your input has been instrumental, and the knowledge I have gained from collaborating with you has profoundly deepened my understanding of the field.

I am thankful to Qualcomm for believing in the potential and real-world applicability of our research and honoring us with the Qualcomm Innovation Fellowship. I am deeply thankful to my mentors, Diyan Teng and Rashmi Kulkarni, for their insightful and constructive feedback, as well as their detailed guidance throughout our projects.

During the summers, I was fortunate to intern with incredible teams at Amazon Web Services and Nokia Bell Labs. I am deeply grateful to my internship managers and mentors for believing in my abilities and entrusting me with critical projects. These internships allowed me to tackle complex challenges and apply technical solutions to real-world problems, broadening my perspective on the impact of our work. I will always cherish these experiences as essential to my growth as a professional, and they have played a pivotal role in shaping my career.

To my beloved wife, Rituparna Datta, I am endlessly grateful for your presence in my life, as your confidence in me has inspired me to reach new heights and continuously strive to be my best self. You have been my unwavering companion, always understanding, even when my academic commitments required me to be away. Together, I look forward to a future filled with shared achievements, joyful moments, and the continued growth of our dreams.

I want to express my deepest gratitude to my family for their unwavering love, support, and the profound impact they’ve had on my life. To my father, Sagar Roy Chowdhury, who has been my moral compass, instilling in me the values of hard work and perseverance; to my mother, Lipika Roy Chowdhury, whose countless sacrifices have been instrumental in shaping both my upbringing and my career; to my grandmother, Kalyani Roy Chowdhury, who, though she could not witness the culmination of my Ph.D. journey, had a transformative influence on my early years; and to my sister, Kaberi Roy Chowdhury, who was integral to my childhood, growing up alongside me and creating cherished memories. I am profoundly grateful for each of you and the vital roles you've played in my life.

I am also grateful to my co-authors for graciously approving the inclusion of the following publications and material in my dissertation:

Chapter 2 incorporates material from the publication "PrimeNet: Pre-training for Irregular Multivariate Time-Series", by Ranak Roy Chowdhury, Jiacheng Li, Xiyuan Zhang, Dezhi Hong, Jingbo Shang, Rajesh K. Gupta, published in Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI 2023). The dissertation author was the primary investigator and the lead author of this paper.

Chapter 3 incorporates material from the publication "Task-Aware Reconstruction for Time-Series Transformer", by Ranak Roy Chowdhury, Xiyuan Zhang, Dezhi Hong, Rajesh Gupta, Jingbo Shang, published in Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2022). The dissertation author was the primary investigator and the lead author of this paper.

Chapter 4, in part is currently being prepared for submission for publication of the material. Chowdhury, Ranak Roy; Kapila, Ritvik; Panse, Ameya; Zhang, Xiyuan; Teng, Diyan; Kulkarni, Rashmi; Hong, Dezhi; Gupta, Rajesh K.; Shang, Jingbo. The dissertation author was the primary investigator and the lead author of this paper.

VITA

2018	B.Sc.Engg. in Computer Science and Engineering, Bangladesh University of Engineering and Technology
2022	M.S. in Computer Science, University of California San Diego
2024	Ph.D. in Computer Science, University of California San Diego

PUBLICATIONS

Xiyuan Zhang, **Ranak Roy Chowdhury**, Rajesh Gupta, Jingbo Shang, "Large Language Models for Time Series: A Survey", In the 33rd International Joint Conference on Artificial Intelligence, IJCAI 2024, Survey Track

Ranak Roy Chowdhury, Ritvik Kapila, Ameya Panse, Xiyuan Zhang, Diyan Teng, Rashmi Kulkarni, Dezhi Hong, Rajesh K. Gupta, Jingbo Shang, "ZeroHAR: Contextual Knowledge Augments Zero-Shot Wearable Human Activity Recognition", (*Under Submission*)

Xiyuan Zhang, **Ranak Roy Chowdhury**, Dezhi Hong, Rajesh Gupta, Jingbo Shang, "UniMTS: Unified Pre-training for Motion Time Series", (*Under Submission*)

Han Guo, Ramtin Hosseini, Ruiyi Zhang, Sai Ashish Somayajula, **Ranak Roy Chowdhury**, Rajesh K. Gupta, Pengtao Xie, "MLO-MAE: Downstream Task Guided Masking Learning in Masked Autoencoders Using Multi-Level Optimization", (*Under Submission*)

Ranak Roy Chowdhury, Jiacheng Li, Xiyuan Zhang, Dezhi Hong, Jingbo Shang, Rajesh K. Gupta, "PrimeNet: Pre-training for Irregular Multivariate Time-Series", In the 37th AAAI Conference on Artificial Intelligence, AAAI 2023

Xiyuan Zhang, Xiaohan Fu, Diyan Teng, Chengyu Dong, Keerthivasan Vijayakumar, Jiayun Zhang, **Ranak Roy Chowdhury**, Junsheng Han, Dezhi Hong, Rashmi Kulkarni, Jingbo Shang, Rajesh Gupta, "Physics-Informed Data Denoising for Real-Life Sensing Systems", In Proceedings of the 21st ACM Conference on Embedded Networked Sensor Systems, SenSys 2023

Ranak Roy Chowdhury, Anshu Bhatia, Haoqi Li, Srikanth Ronanki, Sundararajan Srinivasan, "MARS: Self-supervised Multi Task Pre-training for Accent Robust Speech Representation", (*Under Submission*)

Xiyuan Zhang, **Ranak Roy Chowdhury**, Jiayun Zhang, Dezhi Hong, Rajesh Gupta, Jingbo Shang, "Unleashing the Power of Shared Label Structures for Human Activity Recognition", In Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, CIKM 2023

Xiyuan Zhang, **Ranak Roy Chowdhury**, Jingbo Shang, Rajesh Gupta, Dezhi Hong, "Towards Diverse and Coherent Augmentation for Time-Series Forecasting", In ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2023

Ranak Roy Chowdhury, Xiyuan Zhang, Dezhi Hong, Rajesh Gupta, Jingbo Shang, "Task-Aware Reconstruction for Time-Series Transformer", In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2022

Shuheng Li, **Ranak Roy Chowdhury**, Jingbo Shang, Rajesh K. Gupta, Dezhi Hong, "UniTS: Short-Time Fourier Inspired Neural Networks for Sensory Time Series Classification", In Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems, SenSys 2021

Ranak Roy Chowdhury, Dezhi Hong, Rajesh K. Gupta, Jingbo Shang, "RIoT: Towards Robust Learning for Internet-of-Things", (*Under Submission*)

Xiyuan Zhang, **Ranak Roy Chowdhury**, Jingbo Shang, Rajesh Gupta, Dezhi Hong, "ESC-GAN: Extending Spatial Coverage of Physical Sensors", In Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, WSDM 2022

Ranak Roy Chowdhury, Abdullah Adnan, Rajesh Gupta, "Real Time Principal Component Analysis", ACM Transactions on Data Science, Volume 1, Issue 2, Article No.: 10, Pages 1 - 36, TDS 2020

Ranak Roy Chowdhury, Abdullah Adnan, Rajesh Gupta, "Real Time Principal Component Analysis", In Proceedings of the 35th IEEE International Conference on Data Engineering, ICDE 2019

ABSTRACT OF THE DISSERTATION

Efficient Learning for Sensor Time Series in Label Scarce Applications

by

Ranak Roy Chowdhury

Doctor of Philosophy in Computer Science

University of California San Diego, 2024

Professor Rajesh K. Gupta, Chair
Professor Jingbo Shang, Co-Chair

Sensors in daily life span applications from healthcare and climate modeling to home automation and robotics. These sensors generate abundant time series data, aiding our understanding of various real-life processes. However, much of this data is "unlabeled", that is, without any annotations as to what the data itself refers to in an application. Also, unlike images or text, time series data is challenging for humans to easily interpret and label. Unlike identifying, say objects in a picture, time-series data often requires human experts to analyze using tools and interpret such data before any useful labels can be attached to the data. This is expensive, especially in real-time settings. It is also not scalable due to costs and scarcity of human annotators. Further, labeling data involving human subjects can compromise privacy and security, resulting in a further shortage of labeled data.

Self-supervised learning, that is algorithms that learn from unlabeled data, has been used to address the label scarcity problem by automatically generating pseudo-labels from data itself. However, when applied to sensor data, these algorithms are suboptimal because they have not been specifically adapted or customized for the sensory domain.

In this dissertation, we will develop methods to adapt traditional self-supervised learning algorithms for sensory domain-specific information to mitigate label scarcity issue. Our approach consists of three steps that can be applied progressively with increased effectiveness. First, we integrate time-interval information of unlabeled data into self-supervised algorithms to build a pre-trained model. Next, we fine-tune the pre-trained model by incorporating application-specific knowledge into a self-supervised algorithm that improves the fine-tuning process. Finally, we propose a sensor context-aware self-supervised algorithm to enhance classical fine-tuning that generalizes to novel classes during testing.

We conduct extensive experiments across various sensory data domains, including Motion, Audio, Electroencephalogram (EEG), and Human Activity Recognition (HAR), comparing our methods to leading statistical and deep learning models. By adapting self-supervised algorithms to sensory data with time-awareness, task-specificity, and sensor context-awareness, our methods improve few-shot learning by 10%, fully-supervised learning by 3.6%, and zero-shot learning by 20% compared to the best baselines. Our framework demonstrates state-of-the-art performance across sensing systems of various scales, from small-scale personal healthcare monitoring, human action recognition, and smart home automation to large-scale smart building control, smart city planning, climate modeling, and beyond.

Chapter 1

Introduction

1.1 Growth of Sensing Technologies

Over the past decade, the integration of sensors into our daily lives has significantly increased, driven by the rising demand for applications like healthcare monitoring, smart home automation, high-accuracy motion capture in video games, and the growing adoption of green energy technologies. IoT Analytics predicts that by 2027, there will be over 29 billion IoT connections [1]. According to Statista, revenue in the IoT market in the United States is at 343 billion in 2024 and is projected to increase by approximately $+60\%$ to reach 545 billion in 2028 [2]. Figure 1.1 illustrates the trend of global active IoT connections and the revenue across various sub-sectors under IoT, including Automotive, Heathcare, Consumer, Industrial, Smart city, Smart Finance, and others.

With the rapid expansion of sensor technology, its integration into our daily lives has become crucial. These sensors are fundamental in automating and improving decision-making processes, spanning from personal applications to large-scale industries like agriculture and advanced manufacturing. The vast amounts of data generated by these billions of sensors, often in the form of time series, hold immense potential. Applying machine learning is essential for extracting meaningful insights from this data, allowing us to uncover the underlying dynamics

[1]https://iot-analytics.com/number-connected-iot-devices/
[2]https://www.statista.com/outlook/tmo/internet-of-things/united-statesrevenue

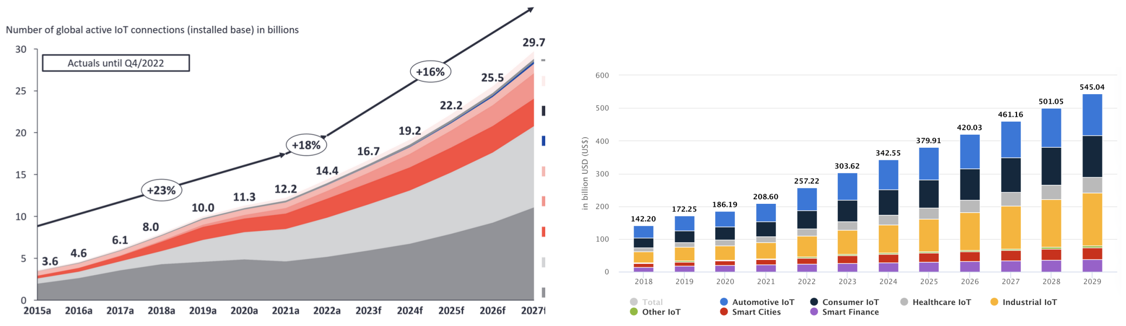


Figure 1.1. The number of sensors covering diverse application domains, as well as their revenue has significantly increased over the years.

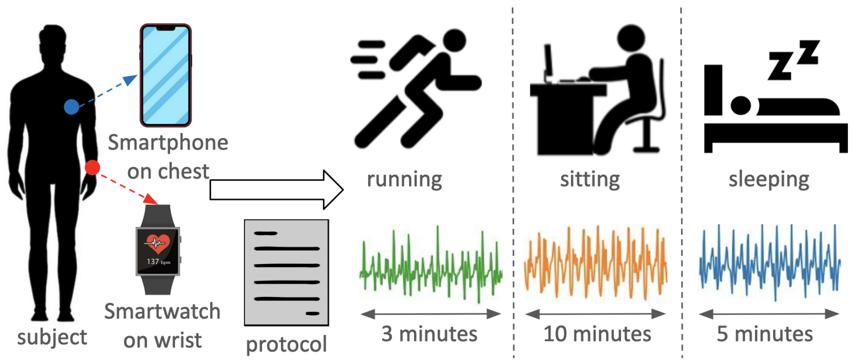


Figure 1.2. Subject follows predefined protocol.

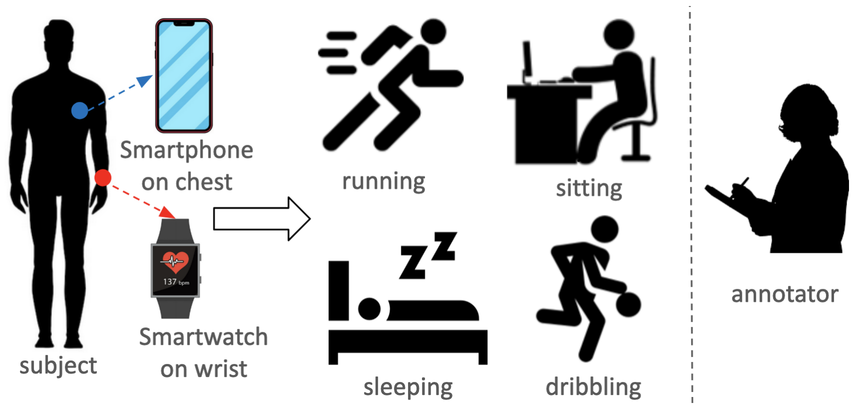


Figure 1.3. Subject conducts activities in-the-wild and a human annotator watches the subject and labels the activities.

and patterns of the physical world and transform raw information into actionable knowledge.

1.2 The Label Scarcity Problem

Despite the wealth of information captured by these sensors, a significant challenge arises: most standard machine learning algorithms rely on labeled data to learn effectively. Unfortunately, the majority of this sensory data is "unlabeled", making it difficult for traditional models to extract value. To address this issue, we have two main approaches: a) either label the vast amounts of unlabeled data or b) design algorithms that can learn directly from unlabeled data.

The first approach is hard for several reasons. Unlike, images, videos or text, it is challenging for humans to easily interpret and understand sensory time-series data by looking at it. Hence, real-world sensory data is difficult to label. Consider training a Human Activity Recognition (HAR) system from wearable sensor data. Billions around the world carry smartphones and smartwatches which are recording their motion data. However, it is difficult to recognize the activity being conducted just by looking at the wearable data, for example, like the curves shown in Figure 1.2. Without the label, this sea of wearable sensing data cannot be harnessed to train a HAR system.

Most of the available labeled data in IoT is collected through experiments under two different settings: 1) controlled experimental settings where subjects operate within a set of predefined rules, as shown in Figure 1.2. For example, to collect labeled wearable sensing data to train a HAR system, we define a set of protocols that a human subject follows, like 3 minutes of running, followed by sitting for 10 minutes, followed by 5 minute of sleeping. 2) Subjects conduct activities in-the-wild and a human annotator monitors the subject in real-time to label the data, as shown in Figure 1.3. For example, we allow the subject to go about his daily activities as usual and make the subject or a different person watch the subject in real-time to annotate the motion data.

Undertaking experiments to collect large-scale sensing data involves high costs from purchasing and deploying high quality, precise sensors, paying human subjects to participate in studies, paying human annotators to watch subjects and label data in real-time. Additionally, experimental data is often less effective than real-world data because it may not capture the full complexity and variability of real-world scenarios. External intervention during experiments can distort the data because subjects, aware of being observed and recorded, may not act naturally, resulting in less robust and generalizable data. Moreover, experiments involving recording and watching human subjects raise security and privacy concerns. These challenges underscore the difficulty of labeling the vast amounts of existing real-world sensory data and the complexities involved in conducting real-time data labeling experiments.

The second approach to counter the label scarcity problem involves designing learning algorithms that can learn directly from unlabeled data. This allows us to leverage the vast amounts of existing real-world sensory data that are naturally collected. Using real-world data is not only cheaper but more effective for training models because it captures authentic, diverse scenarios, leading to more accurate and generalizable models.

1.3 Self-Supervised Learning (SSL)

Self-Supervised Learning (SSL) [29, 15] is a standard technique to learn from data without requiring labeled training examples. Instead, the data provides the supervision. The key idea is to generate supervisory signals from the data, creating pseudo-labels, which the model can use to learn useful representations or features.

SSL involves designing pretext tasks where the model is trained to predict part of the data from other parts. These tasks are designed to encourage the model to learn meaningful features that can be useful for downstream tasks. Standard pretext tasks include masked data reconstruction [46, 16] (predicting missing parts of an image), next token prediction [11, 21] (predicting the next word in a sentence), or contrastive learning [47, 117] (bringing similar data close while pushing away distinct data).

SSL has been widely explored in different domains, including natural language processing

[138, 76], computer vision [44, 46], speech and audio processing [102, 53], time series [131, 35], bioinformatics [96, 93], graph learning [123, 132], autonomous driving [110, 43], reinforcement learning [136, 41], robotics [67, 1] and others.

Most SSL methods for time-series [144, 38] use standard techniques, like masked data reconstruction or contrastive learning, from natural language processing and directly apply them for time-series data. Some works, like [143, 118], use properties like similarity within temporal neighborhood, to modify the existing SSL techniques for time-series.

However, images, text, audio, and standard time-series data differ significantly from sensory time-series data in their structure and characteristics. Images rely heavily on spatial coherence and visual patterns, where each pixel's position and color contribute to the overall visual representation [62, 3], which do not translate well to the irregularities and context-dependencies of sensor data. Text data is sequential but based on discrete symbols (words or characters) with semantic meaning [91, 4], making it inherently different from the raw, continuous signals of sensor data. Audio data is also sequential and continuous, often representing variations in sound over time, but it typically has a more uniform sampling rate and less complex temporal dependencies than sensory data [82, 8]. Standard time-series data, like financial or weather data, usually has a clear and consistent temporal structure [39, 33]. Self-supervision techniques for standard time-series data might not account for the multi-modal, context-rich nature of sensory data [99, 129]. Therefore, self-supervised algorithms developed for images, text, audio, and standard time-series data often fall short when applied to sensory time-series data due to their inability to capture the unique challenges of this domain.

In contrast, sensory time-series data, such as data from accelerometers, gyroscopes, or other sensors, are often characterized by irregular time interval between successive data points, asynchronicity across multiple features, contextual knowledge about sensors and the environment in which they operate [24, 23]. It includes not only the raw measurements but also requires understanding the context, such as the sensor's position and the specific physical activity being monitored, making it more complex and less predictable.

Existing self-supervision methods do not adequately address the need for context-awareness, time-awareness, and task-specific adjustments that are crucial for accurately interpreting and generalizing from sensor data. Leveraging the unique properties of sensory data can significantly enhance the effectiveness of standard self-supervision algorithms for sensory applications.

1.4 Learning Pipeline

In this dissertation, we develop methods to adapt traditional self-supervised learning algorithms to sensory data using domain-specific information to mitigate label scarcity issues. We identify common sensory label constraint scenarios that arise at each of the three stages of model development in a typical machine learning workflow: pre-training, fine-tuning and zero-shot learning.

- **Pretraining** [66, 120]: This stage involves training the model on a large, diverse dataset, often unrelated to the specific task we ultimately want to solve. The goal is for the model to learn general patterns, features, and representations that can be broadly applied. During pretraining, the model develops a strong foundation by understanding the underlying structure of the data, such as identifying edges and textures in images or semantic relationships in text. This process often involves unsupervised or self-supervised learning techniques, especially when labeled data is scarce.

- **Finetuning** [95, 113]: After pretraining, the model is adapted to a specific task or domain by continuing training on a smaller, more focused dataset that is directly related to the desired application. Finetuning allows the model to refine and adjust its learned features to better suit the particularities of the task at hand, such as detecting specific objects in images, recognizing certain activities from sensor data, or understanding domain-specific language. This stage typically uses supervised learning, with labeled data guiding the model to improve its performance on the specific task.

- **Zero-Shot Learning** [135, 75]: In this stage, the model is tasked with recognizing or making predictions about new, unseen classes or categories that were not included in the training data. Zero-shot learning relies on the model's ability to generalize from its prior knowledge, often by leveraging relationships or similarities between the seen and unseen classes. For example, the model might use semantic information, such as descriptions or attributes, to infer the characteristics of the new classes and make accurate predictions without needing additional training data for those classes. This stage is particularly useful when it's impractical or impossible to gather labeled data for every possible class or scenario.

1.5 Human Activity Recognition (HAR)

We present our label-efficient sensor time-series analysis framework in light of one of the more popular sensory application, Human Activity Recognition (HAR). HAR is the process of identifying and classifying physical activities performed by individuals based on data collected from various sensors [57, 54]. HAR is widely used in healthcare, fitness, smart homes, security, workplace safety, transportation, gaming, retail, education, and personal assistance to monitor, analyze, and enhance daily activities and behaviors through sensor data [68, 125].

HAR can be achieved using different types of data sources, including inertial measurement units (IMUs) or wearable sensors, which capture motion data such as acceleration and rotation [5, 56]. Alternatively, HAR can be performed using images [108, 124] or videos [121, 84] from cameras, which provide visual information about the activities. Audio signals can also be used to recognize activities by analyzing sound patterns associated with different actions [85, 111]. Additionally, HAR can leverage Wi-Fi signals to detect movement and infer activities based on signal disruptions and patterns [127, 17]. Another modality for HAR is Electromyography (EMG) data, which captures muscle activity patterns, enabling precise recognition of physical activities, particularly in rehabilitation and prosthetics control [107, 83].

Table 1.1. Metric and directionality of IMU sensors.

IMU	Accelerometer	Gyroscope	Magnetometer
Metric	Linear Acceleration	Angular Velocity	Magnetic Field Strength
X-axis	Left-Right	Forward-Backward	East-West
Y-axis	Up-Down	Left-Right	North-South
Z-axis	Forward-Backward	Clockwise-Counterclockwise	Upward-Downward

Wearable HAR is often considered the most effective compared to other modalities because it directly captures movement data from the body, providing precise and continuous monitoring without the need for external infrastructure. Unlike cameras, wearables do not raise privacy concerns and can function in any environment, regardless of lighting or line of sight issues [30, 63]. Unlike audio-based recognition, wearables are not affected by background noise or the need for clear sound [25, 88]. Wi-Fi-based HAR, while convenient, can suffer from signal interference and requires a controlled environment, whereas wearables offer consistent performance regardless of external conditions [140, 40]. Moreover, wearables can be worn comfortably throughout the day, providing rich, high-resolution data from sensors like accelerometers, gyroscopes, and heart rate monitors, making them highly reliable for a wide range of applications, from health monitoring to fitness tracking. This combination of accuracy, privacy, and environmental independence makes wearable HAR the best choice among the available modalities.

HAR utilizes the motion sensors embedded in everyday smart devices like smartphones and smartwatches, which people typically carry on their bodies. These devices, equipped with inertial measurement units (IMUs) sensors. IMU sensors like accelerometers, gyroscopes, and magnetometers, measure linear acceleration, angular velocity, and magnetic field strength, respectively, along the x-, y-, and z-axes. By analyzing this sensory data, collected from various body positions, machine learning models can be trained to accurately predict physical activities such as walking, running, or sitting. The effectiveness of HAR lies in its ability to interpret the diverse signals generated by these Inertial Measurement Unit (IMU) sensors, making it a valuable

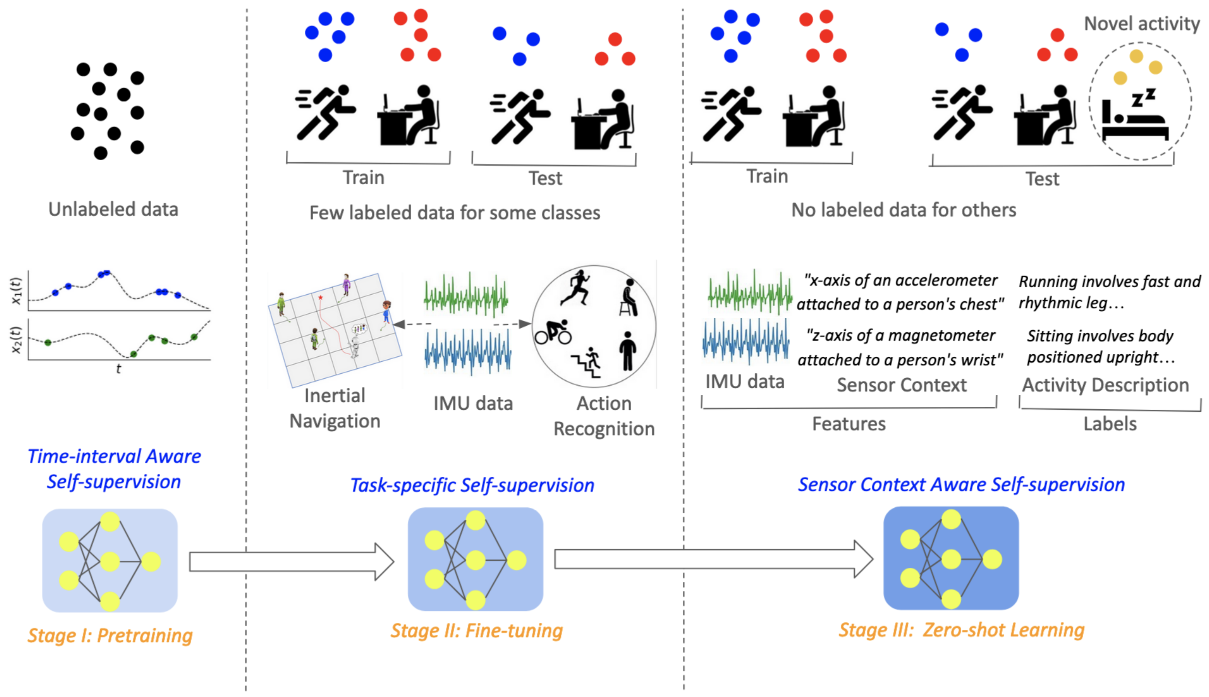

Unlabeled data
$x_1(t)$
$x_2(t)$
t
Time-interval Aware Self-supervision
Stage I: Pretraining
Train
Test
Few labeled data for some classes
Inertial Navigation
IMU data
Action Recognition
Task-specific Self-supervision
Stage II: Fine-tuning
Novel activity
Train
Test
No labeled data for others
"x-axis of an accelerometer attached to a person's chest"
"z-axis of a magnetometer attached to a person's wrist"
Running involves fast and rhythmic leg…
Sitting involves body positioned upright…
IMU data
Sensor Context
Activity Description
Features
Labels
Sensor Context Aware Self-supervision
Stage III: Zero-shot Learning

Figure 1.4. Pipeline of our proposed label-efficient sensor time series analysis framework.

tool for applications in health monitoring, fitness tracking, and beyond. Table 1.1 outlines the different types of IMU sensors, the metric they record and the directionality of those metrics captured by different axes.

1.6 Proposed Framework

Figure 1.4 illustrates our label-efficient learning workflow in the context of HAR for the three stages of learning pipeline. In Stage I, we use the time-interval property of sensory data to adapt standard self-supervised algorithms to pre-train a model [23] (Chapter 2). In the next stage, we use a small proportion of labeled data to train a task-specific self-supervised algorithm that fine-tunes the model from Stage I [24] (Chapter 3). Finally, we consider the scenario where some of the activity labels encountered in real-world may not be present in the training set. So we design a sensor context guided self-supervised algorithm that can identify novel, unseen classes without being trained on them (Chapter 4).

Our methods build on established self-supervised learning techniques, such as masked

Table 1.2. Summary of contributions of our proposed data-efficient learning scheme for label scarce sensory applications.

Stage	I	II	III
Type of Learning	Pre-training	Fine-tuning	Zero-shot Learning
Label Constraints	No labeled data	Few labeled data	No labeled data for some classes
Self-supervision	Data Reconstruction, Contrastive Learning	Data Reconstruction	Contrastive Learning
Domain Property	Time-interval	Task-specificity	Sensor Context
Model	PrimeNet	TARNet	ZeroHAR
Time-series Application	Classification, Regression, Anomaly Detection, Interpolation	Classification, Regression, Anomaly Detection, Interpolation	Classification
Chapter	2	3	4

data reconstruction and contrastive learning, and tailor them specifically for sensory domains by incorporating unique sensory properties. Our label-efficient sensory learning workflow is versatile and can enhance a wide range of applications, from smart buildings and smart cities to healthcare, climate modeling, robotics, and beyond.

1.7 Contribution and Outline

We summarize our contributions and applications of the proposed framework for label-efficient sensory learning in Table 1.2. The technical contributions in this dissertation are supported by specific implementation tools that are available on the open-source Github repository as described in the Abstract for each chapter. More specifically,

- **Time-interval Aware Self-supervision** [23] (Chapter 2) is the first representation learning pre-trained model for irregular multivariate time-series. Many practical sensory applications involve irregular and asynchronous time series, for which the time intervals between successive observations are non-uniform and different features are sampled at different times. We use these time intervals as signals to design a time-sensitive contrastive learning and data reconstruction task to pre-train a model from unlabeled data. Experiment results show that our method significantly outperforms state-of-the-art methods on naturally occurring irregular and asynchronous data from Healthcare and IoT applications for several downstream tasks, including classification, interpolation, and regression.

- **Task-specific Self-supervision** [24] (Chapter 3) fine-tunes the pre-trained model from Chapter 2. During fine-tuning, it augments existing fully-supervised approaches for limited labels by incorporating task-aware data reconstruction as an auxiliary task to the end task. Recent studies suggest that learning a self-supervised representation from labeled data before tackling the end task improves performance. However, because these tasks are done sequentially, the self-supervised learning remains task-agnostic. Our approach runs these tasks in parallel, allowing labels from the end task to inform and enhance the self-supervised learning, making it more task-aware. We use self-attention score distribution from end-task training to identify timesteps deemed important by the end task. We then uses those timesteps to design a data-driven masking strategy for a data reconstruction task. Extensive experiments on tens of classification and regression datasets show that our method results in marked improvement compared to state-of-the-art baseline models across all evaluation metrics under limited labeled data.

- **Sensor Context Aware Self-supervision** (Chapter 4) further refines the model from Chapter 3 for zero-shot learning. Standard fully-supervised models assume that the training data contains all types of labels that a model may encounter after deployment. Our method breaks away from this assumption and designs a model that can identify novel classes despite not being trained on them. We exploit sensor name and placement information, which provide rich contextual information about sensors in complement with motion data to contrastively align them. Finally, this model is trained further with activity descriptions for classification. Experiments on datasets, containing a wide range of sensors, placements and activities show that our method achieves remarkable performance improvement in zero-shot accuracy and macro-f1.

Overall, this dissertation effectively addresses the challenges of learning under different scenarios of label scarcity in sensor time series, significantly improving the performance of sensor time series analysis systems under label deficiency. We show that sensory applications can be improved by a) better utilizing the large corpus of unlabeled data *(Time-interval Aware Self-supervision: Chapter 2)*, b) maximizing performance out of the limited labeled data *(Task-specific Self-supervision: Chapter 3)*, and c) transferring knowledge from the data for available classes to improve performance on novel, unseen classes *(Sensor Context Aware Self-supervision: Chapter 4)*. The resulting framework has wide applications across sensing systems of various scales, from small-scale personal healthcare monitoring, smart home automation, to large-scale smart building control, energy management, climate modeling and beyond.

Chapter 2

Time-interval Aware Self-supervision

Real-world applications often involve *irregular* time series, for which the time intervals between successive observations are non-uniform. Irregularity across multiple features in a multi-variate time series further results in a different subset of features at any given time (i.e., *asynchronicity*). Existing pre-training schemes for time-series, however, often assume regularity of time series and make no special treatment of irregularity. We argue that such irregularity offers insight about domain property of the data—for example, frequency of hospital visits may signal patient health condition—that can guide representation learning.

In this work, we propose PrimeNet [1] to learn a self-supervised representation for irregular multivariate time-series. Specifically, we design a time-sensitive contrastive learning and data reconstruction task to pre-train a model. Irregular time-series exhibits considerable variations in sampling density over time. Hence, our triplet generation strategy follows the density of the original data points, preserving its native irregularity. Moreover, the sampling density variation over time makes data reconstruction difficult for different regions. Therefore, we design a data masking technique that always masks a constant time duration to accommodate reconstruction for regions of different sampling density. We learn with these tasks using unlabeled data to build a pre-trained model and fine-tune it on downstream tasks with limited labeled data, in contrast with existing fully supervised approach for irregular time-series, requiring large amounts of labeled data. Experiment results show that PrimeNet significantly outperforms state-of-the-art

[1] Code is publicly available at https://github.com/ranakroychowdhury/PrimeNet

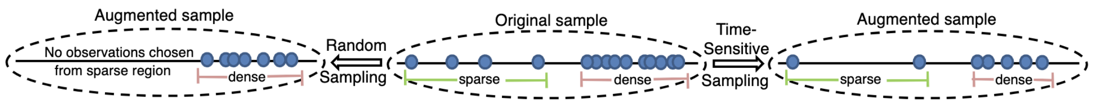
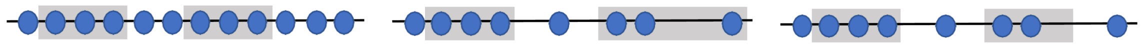


Figure 2.1. Augmenting time-series through different sampling techniques.



(a) Constant length masking for regular time-series.
(b) Constant length masking for irregular time-series.
(c) Constant time duration masking for irregular time-series.

Figure 2.2. Difference between constant time and constant length masking.

methods on naturally irregular and asynchronous data from Healthcare and IoT applications for several downstream tasks, including classification, interpolation, and regression.

2.1 Problem Statement

Many real-world applications generate data with non-uniform time-interval between successive observations. For example, sensors are triggered at irregular intervals driven by events in real-life. Further, not all sensors are triggered at the same time. Thus, irregularity (in time) and asynchronicity (across sensors) are natural characteristics of many time series that provide rich insights into real-world events.

Naturally occurring irregularity in many datasets reflect intrinsic domain property about the underlying system, which can be leveraged to learn the task better. For example, to predict whether a patient is sick or healthy based on their schedule of doctor visits and medical test results, the frequency of visits may be a useful signal, in addition to the physiological variables, i.e. sick patients visit doctors more frequently than healthy ones. However, no regular time series models, whether fully- [24], semi- [144] or un- [118] supervised, encode time information. They assume constant time intervals between all consecutive observations in a sequence (*regularity*) with all features observed at any given time (*synchronicity*) [80]. Hence, simply adapting pre-trained regular time-series methods for irregular time-series is sub-optimal.

Recently, ODE- [100], attention- [105], and set- [50] based models that directly learn time

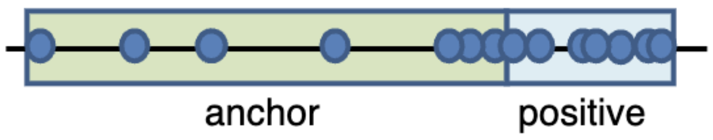


Figure 2.3. Subsampling from irregular time-series results in non-representative samples.

information and encode irregularity have outperformed regular time series models on irregular data. However, most of these irregular methods are fully- [100] or semi- [105] supervised, requiring large amounts of labeled data. Data labeling in IoT is time-consuming and both cost and labor-intensive as it involves physical sensor deployment and human annotators with domain expertise. Moreover, accessing labeled data in healthcare may raise security and privacy concerns.

To this end, we propose PrimeNet, the first pre-trained model for irregular multivariate time series. Specifically, we design two time-sensitive tasks based on contrastive learning and data reconstruction to build a self-supervised representation from *completely unlabeled* irregular time series data.

Time-slicing [38], which chunks a time series into slices containing equal number of readings, is commonly used to augment regular time-series data for contrastive learning. However, it cannot generate representative sub-sequences for an irregular time series as irregular time series exhibit significant variations in sampling density over time and the sampling density of a given time slice may not mirror that of the entire time series. For an irregular time-series, time-slicing the sequence generates an anchor and a positive, neither of which reflects the sampling density variation of the original data (Figure 2.3).

Figure 2.1 shows augmented subsamples resulting from different sampling techniques. The solid black line represents the time axis and an blue circle represents an observation in time. A common sampling approach would be to randomly sample observations from an irregular time series to construct a sub-sequence. However, an irregular time series may also exhibit significant imbalanced occurrences of dense and sparse observations. Hence, random sampling will lead to high sampling bias, drawing observations from some regions but not others, resulting in an unrepresentative sub-sequence. Our time-sensitive stratified sampling draws observations from both dense and sparse regions, better reflecting the sampling density variation of the original sample.

Instead, we propose a time-sensitive stratified sampling technique that draws observations from dense and sparse regions of the time series and combine them to construct a sub-sequence. The generated sub-sequences better reflect the sampling density variations and are, therefore, more representative of the original data.

Masking and reconstructing observations learn a good representation for regular time-series [144]. Figure 2.2 shows how data points within a time-series are typically masked. Grey box shows the time duration of successive observations we mask under each masking technique. For a regular time series, masking the same number of successive observations for every segment (fixed *length*) will always mask over the same time duration of data because the sampling density is constant (Figure 2.2(a)). However, for an irregular time series with inconsistent sampling density, a fixed length mask segment will mask over a different duration for different segments depending on their respective sampling densities (Figure 2.2(b)). Hence, the difficulty to reconstruct data varies through an irregular time series, resulting in poor representations. Therefore, we propose a fixed *time* masking technique that always masks over the same duration of data instead (Figure 2.2(c)). This effectively adjusts the number of masked observations for each segment depending on its sampling density, thereby stabilizing the reconstruction task across regions of different sampling densities, and improving the learned representation.

We conduct experiments on naturally occurring irregular and asynchronous time-series datasets from Healthcare [106, 58], Activity [59], and Energy [115]. We conduct analysis on very few-shot settings and also on full training data settings for several downstream tasks, including classification, regression, and interpolation. We compare PrimeNet to several 1) self-supervised methods for regular time series to show how modeling irregularity helps and 2) fully supervised methods for irregular time series to show the performance boost from using unlabeled data.

Experiment results show that PrimeNet significantly outperforms all baselines on all datasets for all downstream tasks, under both few-shot and full training data settings.

To summarize, our main contributions include:

- We present PrimeNet to learn a self-supervised representation from irregular and asynchronous time series.
- We propose time-sensitive contrastive learning and data reconstruction to learn from data irregularity patterns.
- We evaluate PrimeNet on numerous real-world datasets across several downstream tasks to validate and quantify its efficacy compared with benchmark methods.

2.2 Related Work

2.2.1 Self-Supervised Regular Time Series Methods

Recent research on learning unsupervised representations from *regular* and *synchronous* time-series performs well under limited labeled data settings, using triplet loss [38], hierarchical contrastive loss [142, 22], Fourier transform [73, 146], task-aware reconstruction [24]. They can be adapted to irregular time-series by discretizing continuous-time samples into uniformly-spaced fixed-size bins [104, 147]. However, irregular time series are marked by regions of high and low sampling densities. A wide bin would aggregate data in dense regions, resulting in a loss of fine-grained details. A short bin would generate a high fraction of missing data in sparse regions exploding the sequence length, making imputation increasingly difficult. Consequently, imputation-based methods are hardly used to learn from such irregular time series. Moreover, by treating irregular time series data like a regular one, they abstract the vital irregular time information away from the model, inhibiting performance.

We exploit the irregular time-interval property to design our self-supervsion tasks to pre-train PrimeNet, thereby learning suitable representations from unlabeled *irregular* and *asynchronous* multivariate time-series data.

2.2.2 Irregular Time Series Methods

Gated Recurrent Unit (GRU)-based methods [12] involve modifying the LSTM forget gate [92] or introducing new time gate [87]. [13] combined a neural network with a latent ordinary differential equation (ODE) model. [100, 69] used neural ODEs to model hidden state dynamics. Others use attention mechanism [149, 116] similar to [122] by replacing positional encoding with a fixed time encoding. [105, 61] learns time representation.

These models work under fully- or semi-supervised setting and require large amounts of labeled data to learn a task. PrimeNet leverages unlabeled data to operate completely under self-supervised setting. Once pre-trained, PrimeNet can learn any downstream task with limited labeled data.

2.3 Notation

An individual data case is a N-dimensional, irregularly and asynchronously sampled multivariate time series, $\mathrm{D} = (\mathrm{T}, \mathrm{X}, \mathrm{M})$, where $\mathrm{T} \in \mathbb{R}^{S}$, $\mathrm{X} \in \mathbb{R}^{S \times N}$, and $\mathrm{M} \in \mathbb{R}^{S \times N}$. T denotes the union of timestamps at which all the N features have been sampled and S is the number of such timestamps. Let t and n represent a particular time and feature, respectively, in D. X constitutes a sequence of S feature vectors. $\mathrm{X}_t \in \mathbb{R}^{N}$ represents the feature values sampled at time t. This formulation also covers the uni-variate case when $N = 1$. However, since the time-series is asynchronously sampled, not all the N features may be sampled at t. Hence, we use masking variable M to denote the set of observed features. Specifically, $\mathrm{M}_t \in \mathbb{R}^{N}$ denote the set of observed and unobserved features at t. If feature n is sampled at t, then $\mathrm{M}_{tn} = 1$, otherwise $\mathrm{M}_{tn} = \mathrm{X}_{tn} = 0$. Using M to deal with unobserved dimensions allows us to transform the irregular length time-series into uniform length and parallelize computations, thereby enabling efficient GPU implementation.

Our time-sensitive self-supervision tasks are model agnostic and can be plugged into any irregular time series architectures, like ODE-, attention-, or set-based models, that explicitly encode time information. We follow a similar architecture to mTAND [105] because it is the current state-of-the-art for irregular time series. Our model is pre-trained on unlabeled data using time-sensitive contrastive and data reconstruction loss and then fine-tuned on available labeled data for a given downstream task.

2.4 Model Architecture

We present PrimeNet's core architecture - Time Embedding, Time-Feature Attention (TFA) and Feature-Feature Attention (FFA), as shown in Figure 2.4.

Time Embedding [105] embeds continuous time points into a vector space by leveraging H embedding functions $\phi h(\mathrm{T})$, each outputting a representation of size d_r. Dimension i of embedding h is defined as:

$$\phi h(\mathrm{T})[i] = \begin{cases} \omega_{0h}.\mathrm{T} + \alpha_{0h}, \text{ if } i = 0 \\ sin(\omega_{ih}.\mathrm{T} + \alpha_{ih}), \text{ if } 0 < i < d_r \end{cases} \tag{2.1}$$

where ω_{ih}'s and α_{ih}'s are learnable parameters. Linear term encodes the non-periodic patterns. Periodic terms captures periodicity with ω_{ih} and α_{ih} as frequency and phase.

Time-Feature Attention, TFA [105]

Time-Feature Attention captures the interaction between feature values with their corresponding sampling times. The output from the Time Embedding layer forms the query Q_T and key K_T vectors.

$$\begin{aligned} \mathrm{TFA}(\mathrm{Q}_T, \mathrm{K}_T, \mathrm{M}, \mathrm{X}) &= (\mathrm{M} \odot \mathrm{A}_T)\mathrm{X}, \\ \mathrm{A}_T &= \mathrm{softmax}(\mathrm{Q}_T \mathrm{K}_T / d_r) \end{aligned} \tag{2.2}$$

We do an element-wise multiplication of M with A_T to nullify the effect of unobserved

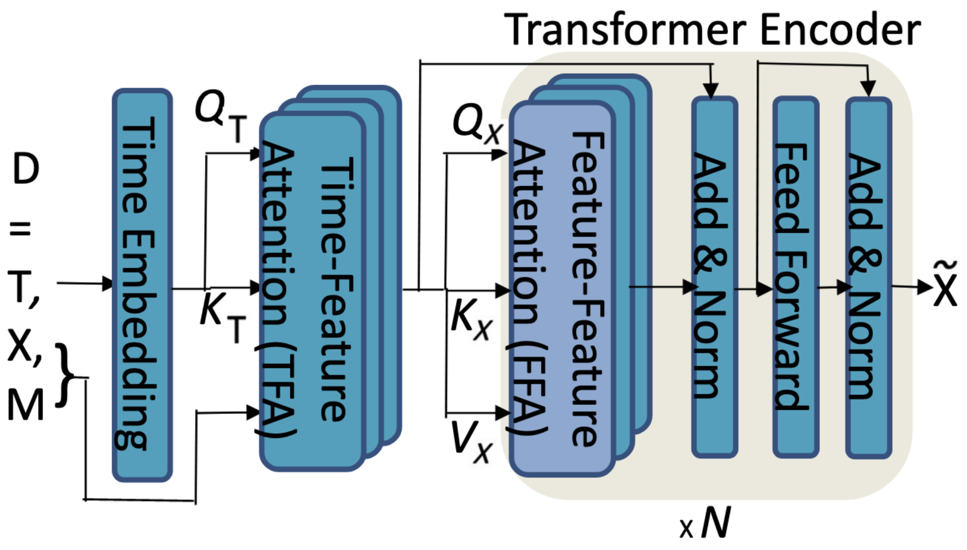


Figure 2.4. PrimeNet Model Architecture Overview.

features in X at a given sampling time in T. We compute a weighted summation of features, where weights are self-attention scores of features' corresponding sampling time embeddings, Q_T and K_T.

Feature-Feature Attention, FFA [122]

Feature-Feature Attention models the self-attention within features. The output from TFA becomes the query, key and value vectors for FFA.

$$Q_X = K_X = V_X = \text{TFA}(Q_T, K_T, M, X) \tag{2.3}$$

Then, FFA will encode the outputs from TFA as follows:

$$\begin{aligned} \text{FFA}(Q_X, K_X, V_X, M) &= (M \odot A)V_X, \\ A &= \text{softmax}(Q_X K_X / d_r) \end{aligned} \tag{2.4}$$

The output of FFA is fed through residual and feed forward layers like in a typical Transformer Encoder. The data is fed through N such Encoders to generate the output $\tilde{X}$.

2.5 PrimeNet Framework

We present two time-sensitive self-supervised pre-training objectives, namely, Time Contrastive Learning (TimeCL) and Time Reconstruction (TimeReco). We discuss how we augment the data, followed by loss computation details.

2.5.1 Time Contrastive Learning (TimeCL)

Augmenting data for contrastive learning through time-slicing or random sampling fails to capture the irregularity in data, motivating our stratified-sampling based approach.

Data Augmentation

Contrastive learning [15, 48] augments observations from the same input to form an anchor and positive (similar to anchor), while a sample from other in-batch inputs form the negative (different from anchor). This pulls the latent space of the anchor and positive closer while pushing away the negatives, learning good data representation [55, 72].

To better replicate the irregularity pattern of data, we maintain an approximate sampling density distribution of the data in its augmented sub-sequences, as outlined in Algorithm 1. For every sampling time T_i, we compute its mean time interval, Z_i, by averaging the time lapse between its immediate predecessor, T_{i-1}, and successor, T_{i+1}, to estimate the local density of the sampled features at T_i (Line 1). We reorder Z and T in ascending order of Z and group T into two bins, T_{dense} (lowest 50% Z-values) and T_{sparse} (highest 50% Z-values) (Lines 2 - 4). We sample a proportion of the timestamps from T_{dense} and the remaining from T_{sparse} to form the sampled timestamps of one augmented sub-sequence, T_A (Line 6). The remaining $T - T_A$ timestamps form the sampled timestamps of the other sub-sequence, T_P (Line 8). This stratified sampling technique ensures that we draw observations from regions of different sampling density of D, regardless of how scarce the observations from certain regions are. Hence, the augmented sub-sequence can better approximate the irregularity in D. To preserve asynchronicity of D, we extract only the subset of features that was sampled together at a given time in D to form

Algorithm 1. TimeCL Data Augmentation

```
Input: $D$
Hyper-parameters: $(\mu_l, \mu_u)$, $(\lambda_l, \lambda_u)$
Output: $D_A$, $D_P$

 1: $\mathrm{Z} \leftarrow \{\frac{(\mathrm{T}_i - \mathrm{T}_{i-1}) + (\mathrm{T}_{i+1} - \mathrm{T}_i)}{2} | \forall i \in \mathbb{Z}^+, 1 < i < S\}, \mathrm{Z} \in \mathbb{R}^S$
 2: Sort $(\mathrm{Z}, \mathrm{T})$ in ascending order of Z
 3: $\mathrm{T}_{dense} \leftarrow \mathrm{T}[1 : S/2], \mathrm{T}_{dense} \in \mathbb{R}^{S/2}$
 4: $\mathrm{T}_{sparse} \leftarrow \mathrm{T}[S/2 : S], \mathrm{T}_{sparse} \in \mathbb{R}^{S/2}$
 5: $\mu \sim \mathrm{U}(\mu_l, \mu_u), 0 < \mu_l < \mu_u < 1$
 6: $\lambda \sim \mathrm{U}(\lambda_l, \lambda_u), 0 < \lambda_l < \lambda_u < 1$
 7: $\mathrm{T}_A \sim$ Sample $\lambda \mu S$ and $(1-\lambda)\mu S$ timestamps from $\mathrm{T}_{dense}$ and $\mathrm{T}_{sparse}$, respectively, $\mathrm{T}_A \in \mathbb{R}^{\mu S}$
 8: $\mathrm{T}_P \leftarrow \mathrm{T} - \mathrm{T}_A$, $\mathrm{T}_P \in \mathbb{R}^{(1-\mu)S}$
 9: $\mathrm{D}_A, \mathrm{D}_P \leftarrow (\mathrm{T}_A, \mathrm{X}_{\mathrm{T}_A}, \mathrm{M}_{\mathrm{T}_A}), (\mathrm{T}_P, \mathrm{X}_{\mathrm{T}_P}, \mathrm{M}_{\mathrm{T}_P})$
10: return $D_A$, $D_P$
```

the feature set for that time in the augmented sub-sequence. Thus, $\mathrm{D}_A = (\mathrm{T}_A, \mathrm{X}_{\mathrm{T}_A}, \mathrm{M}_{\mathrm{T}_A})$ and $\mathrm{D}_P = (\mathrm{T}_P, \mathrm{X}_{\mathrm{T}_P}, \mathrm{M}_{\mathrm{T}_P})$ form good quality anchor and positive sub-sequences, respectively, that are representative of D, thereby improving contrastive learning.

Example 1 Consider the following sampling times for a given sample in seconds:
$T = [2, 5, 12, 16, 17, 19, 21, 22, 24, 26, 27, 34, 38, 47, 53, 60]$
In this sample, we find that the data was sampled at a high frequency from timestamp 16 to 27, but was sampled sparsely before and after these timestamps. To facilitate good representation learning through contrastive learning, the augmented anchor and positive are supposed to be similar to each other and to the original data point.
If we sampled a random continuous sub-sequence from the above data to construct the anchor and positive, then their respective sampling times for the anchor, T_A, and positive, T_P, could be as follows:
$T_A = [17, 19, 21, 22, 24]$, $T_P = [27, 34, 38, 47, 53]$
In the above sampling procedure, the anchor was constructed from the dense part of the original data, while the positive was constructed from the sparse part of the original data. Hence, the resulting anchor and positive are neither similar to one another, nor to the original data point.
The following shows how the anchor and positive will be augmented from the original data point if we follow Algorithm 1.
We first compute the average density metric for each sampled point in the data,
$Z = [3, 5, 5.5, 2.5, 1.5, 2, 1.5, 1.5, 2, 1.5, 4, 5.5, 6.5, 7.5, 6.5, 7]$
Sorting (Z, T) in ascending order of Z, i.e, in order of increasing density,

$Z_{sorted} = [1.5, 1.5, 1.5, 1.5, 2, 2, 2.5, 3, 4, 5, 5.5, 5.5, 6.5, 6.5, 7, 7.5]$
$T_{sorted} = [17, 21, 22, 26, 19, 24, 16, 2, 27, 5, 12, 34, 38, 53, 60, 47]$
Constructing the bins T_{dense} and T_{sparse},
$T_{dense} = [17, 21, 22, 26, 19, 24, 16, 2]$
$T_{sparse} = [27, 5, 12, 34, 38, 53, 60, 47]$
Considering $\lambda = 0.4$ and sampling a sequence of 5 timestamps for anchor and positive, then the respective sampling times for anchor and positive could be as follows:
$T_A = [12, 21, 22, 53, 60]$, $T_P = [5, 19, 24, 38, 47]$
T_A was constructed from timestamps 21 and 22 of T_{dense} and timestamps 12, 53, and 60 of T_{sparse}. Similarly, T_P was constructed from timestamps 19 and 24 of T_{dense} and timestamps 5, 38, and 47 of T_{sparse}. The resulting anchor and positive are now more similar to each other and to the original data point because they have representations from both the dense and sparse parts of the series.

Contrastive Loss, L_{CL}

For a given X_A and X_P, the X_N is formed from all other instances in the same mini-batch. We add a special `[CLS]` symbol in front of every input and pass it through an embedding layer. The final hidden state corresponding to `[CLS]` is used as the aggregate sequence representation for contrastive learning.

We use Normalized Temperature-scaled Cross Entropy (NT-Xent) Loss [15] as our Contrastive Loss function. This has shown improvement over other contrastive losses, like CPC [89] and MoCo [47], in several other domains [15]. It is a modification of the multi-class B-pair loss, where B is the batch size, with addition of the temperature parameter, τ, to scale the cosine similarities as follows:

$$L_{CL} = -\log \frac{\exp(\tilde{\mathrm{X}}_i \tilde{\mathrm{X}}_j / \tau)}{\sum_{k=1}^{2B} \exp(\tilde{\mathrm{X}}_i \tilde{\mathrm{X}}_k / \tau)} \tag{2.5}$$

2.5.2 Time Reconstruction (TimeReco)

Masking constant length of data across time is not suitable to learn reconstruction from non-uniform time-interval data, prompting a constant time masking technique.

Algorithm 2. TimeReco Data Augmentation

Input: D
Hyper-parameters: J, α
Output: D_U, D_V

```
 1: $\mathrm{X}_U, \mathrm{M}_U \leftarrow \mathrm{X}, \mathrm{M}$
 2: $\mathrm{X}_V, \mathrm{M}_V \leftarrow$ initialized with $0's$
 3: for $n$ in $N$ do
 4:     $\mathrm{T}_n = \mathrm{T}[\mathrm{M}_n], t_n \in \mathbb{R}^{L_n}$
 5:     $q_n \leftarrow \alpha(\mathrm{T}_n[-1] - \mathrm{T}_n[0])$
 6:     for $j$ in $J$ do
 7:         $t \sim$ Randomly sample a timestamp from $\mathrm{T}_n$
 8:         $\mathrm{X}_{V_n}[t : t+q_n], \mathrm{M}_{V_n}[t : t+q_n] = \mathrm{X}_{U_n}[t : t+q_n], \mathrm{M}_{U_n}[t : t+q_n]$
 9:         $\mathrm{X}_{U_n}[t : t+q_n], \mathrm{M}_{U_n}[t : t+q_n] = 0, 0$
10:     end for
11: end for
12: $D_U, D_V \leftarrow (\mathrm{T}, \mathrm{X}_U, \mathrm{M}_U), (\mathrm{T}, \mathrm{X}_V, \mathrm{M}_V)$
13: return $D_U$, $D_V$
```

Data Augmentation

Reconstruction for irregular time-series presents two key challenges. First, some regions are more densely sampled than others. Therefore, masking a constant length of data will mask over a shorter time-interval for a denser region compared to a sparser region, as shown in Figure 2.2(b). Hence, reconstructing data at the masked segment may be trivial for the dense region with abundant unmasked data in close temporal vicinity to use as contextual information but difficult for a sparse region. Second, for multivariate time-series, each feature may have different sampling frequency and may be sampled over different duration. Hence, each feature has different time gaps between successive observations, i.e. asynchronicity, rendering a constant length masking strategy ineffective.

To learn a better reconstruction for irregular time-series, we propose Algorithm 2. We specify the number of masking segments J, and the fraction of time interval α, to mask for each segment. To address the asynchronicity problem, we compute the timespan q_n to mask separately for each feature n (Line 5) because the total duration for which each feature lasts may vary. Features lasting shorter should have shorter masking segments as compared to those lasting longer. To deal with irregularity within a time-series, we fix the timespan q_n to mask for feature n, instead of fixing the number of observations to mask. This adapts the length of masking segment based on the sampling density of the time-series in the masking region, as shown in Figure 2.2(c). For a given q_n, a dense region will mask more observations than a sparse region. Hence for denser regions, there will not be sufficient unmasked observations in close temporal proximity of the masked segment to make the reconstruction trivial. Similarly for sparser regions, the number of masked observations will be low, so there will be sufficient observations in the temporal vicinity of the masked region to keep the task tractable. Lines 6 - 10 outlines this procedure.

Example 2 Consider the following sampling times for a given sample in seconds:
$T = [2, 5, 12, 19, 28, 37, 45, 47, 49, 50, 51, 53]$
During masked data reconstruction for a regularly sampled time-series, we mask out multiple contiguous non-overlapping segment of fixed length from different sections of the data and predict the masked out values. Let's consider that for the above timestamps, we mask out 2 segments, each of length 3. Then we could mask out the data corresponding to the following timestamps,
First masked segment $= [5, 12, 19]$
Second masked segment $= [47, 49, 50]$
And the set of unmasked timestamps, T_U, that can be used as context to infer the feature values at the masked out timestamps are,
$T_U = [2, 28, 37, 45, 51, 53]$
As we can see, the first masked segment is derived from a sparse part of the data, while the second masked segment comes from a denser part. It is more difficult to infer the features values for the sparser part because there are too few data points in their temporal vicinity to use as context, i.e. data points sampled at timestamps 2, 29 and 38 are too sparse to be used as context, effectively. By the same logic, it is also much easier to infer the features values for the denser part because there are many data points in their temporal vicinity to use as context, i.e. data points sampled at timestamps 45, 51 and 53 are quite dense to enable easy reconstruction of the masked out segment at timestamps 45, 51 and 53.

The difficulty of reconstructing a particular masked segment is dependent on the number of data points in the masked segment that needs to be reconstructed and the number of data points that are in the temporal vicinity of the masked out segment, since feature values sampled closer in time tend to be more similar to each other than distant ones. Hence, if the number of masked out data points between two segments are equal, then the reconstruction task is easier for the denser parts than for the sparser parts because there are more data points in the temporal vicinity of a masked out segment in the dense region.

Therefore, in order to balance the difficulty of reconstruction between segments of varying density in an irregularly sampled time-series, we must adjust the number of masked out data points in each segment. The length of the masked out segment is adjusted based on the density of the data in the masking region. We want to mask out fewer data points from the sparser part so that fewer data points need to be inferred and there are more data points available in the temporal vicinity of the masked out segment to be used as context. This makes reconstruction easier for the sparser part. Similarly, we want to mask out more data points from the denser part so that more data points need to be inferred and there are more fewer points available in the temporal vicinity of the masked out segment to be used as context. This makes reconstruction difficult for the denser part.

In order to achieve this, instead of masking out fixed *length* segments from different parts of the data, we will mask out fixed *time* segments. Based on Algorithm 2, let's consider that we mask out 2 segments, each of a duration of 5 seconds. Then we could mask out the data corresponding to the following timestamps,
First masked segment between timestamp 10 to 15 = $[12]$
Second masked segment between timestamp 48 to 53 = $[49,50,51,53]$
And the set of unmasked timestamps, T_U, that can be used as context to infer the feature values at the masked out timestamps are,
$T_U = [2,5,19,28,37,45,47,53]$
By masking fixed time segments, instead of fixed length segments, we are masking out fewer data points from the sparser parts and more data points from the denser parts. This also leaves more data points in the temporal vicinity of the masked out segment in the sparser part, while leaving fewer data points in the temporal vicinity of the masked out segment in the denser part. Hence, the reconstruction gets easier for the sparser regions and more difficult for the denser regions. By adjusting the masking length across different segments of the data, we are balancing the difficulty of reconstruction based on density of the masked out segment.

Reconstruction Loss, L_{Reco}

We feed the masked out features X_U to PrimeNet and extract the generated features $\tilde{\mathrm{X}}_U$. The Reconstruction Error between model output $\tilde{\mathrm{X}}_U$ and target X_V, is computed using Mean Squared Error (MSE),

$$L_{Reco} = \left\| \mathrm{M}_V \odot (\tilde{\mathrm{X}}_U - \mathrm{X}_V) \right\|_2^2 \tag{2.6}$$

Hence the total loss L becomes:

$$L = \eta L_{CL} + (1-\eta) L_{Reco}, \tag{2.7}$$

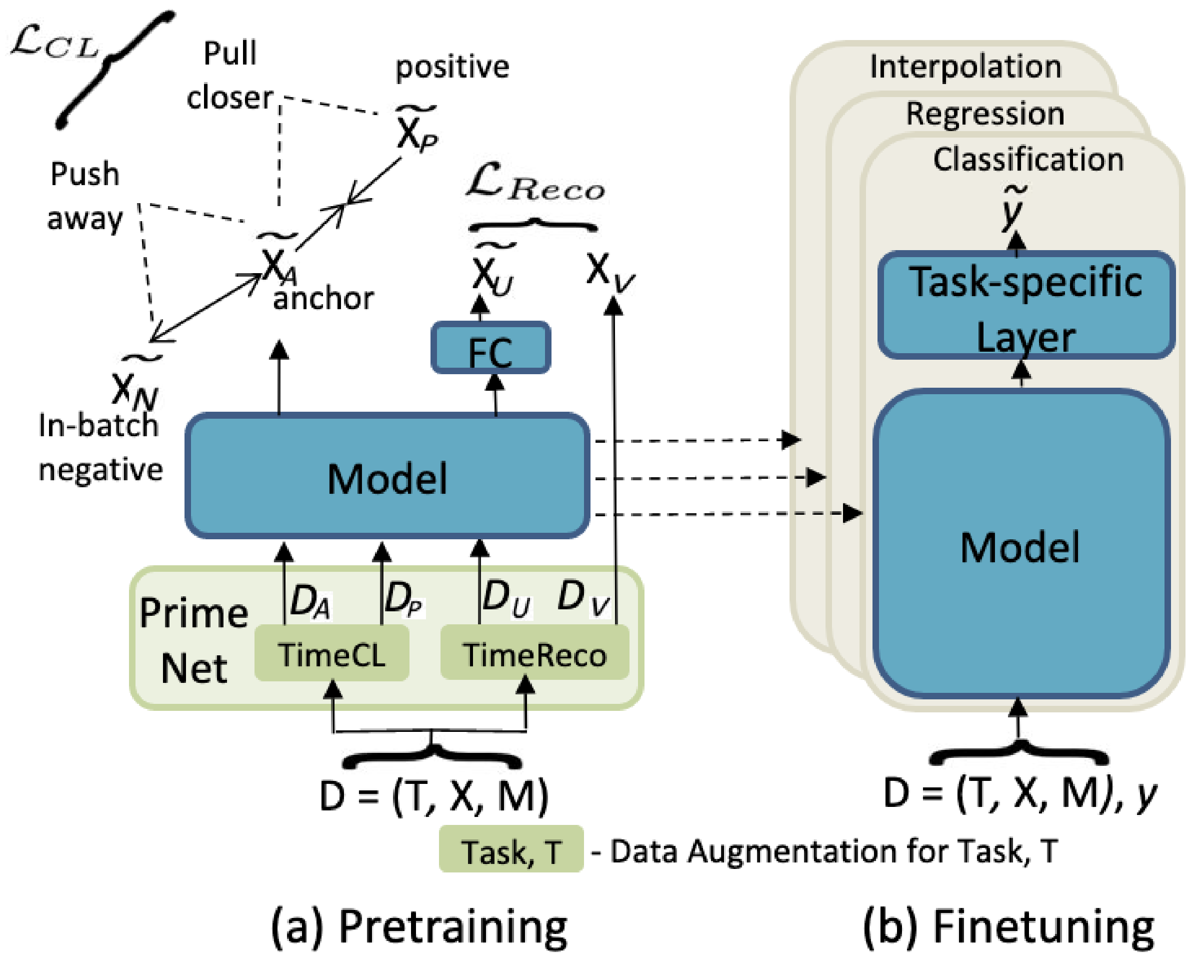


Figure 2.5. PrimeNet Pre-training and Fine-tuning Overview.

where η is a hyperparameter, $0 < \eta < 1$. It balances the two losses because different datasets may benefit differently from these two tasks.

Figure 2.5(a) shows the pre-training workflow of PrimeNet. Green modules represent the data augmentation algorithms. Blue modules represent the learnable components. For a given data point D, we feed it through Algorithm 1 to generate the anchor D_A, and positive D_P. Additionally, we feed D through Algorithm 2 to mask its features and prepare the masked input D_U and target D_V.

For fine-tuning on supervised downstream tasks, like classification, regression and interpolation, we append task-specific layers on top of pre-trained PrimeNet, as shown in

Table 2.1. Classification on PhysioNet [106] measured by Area Under the ROC curve (AUC). Higher AUC is better.

Baselines	1-shot	2-shot	4-shot	Full
	Self-supervised regular time-series methods			
TNC	<u>0.615 ± 0.014</u>	<u>0.641 ± 0.023</u>	<u>0.632 ± 0.011</u>	0.755
TS2Vec	0.554 ± 0.04	0.556 ± 0.046	0.584 ± 0.023	0.737
TST	0.535 ± 0.021	0.564 ± 0.030	0.552 ± 0.042	0.815
	Irregular time-series methods			
mTAND	0.612 ± 0.016	0.587 ± 0.009	0.598 ± 0.009	<u>0.837</u>
GRU-Mean	0.542 ± 0.054	0.523 ± 0.042	0.575 ± 0.038	0.806
P-LSTM	0.579 ± 0.058	0.551 ± 0.048	0.569 ± 0.049	0.782
RNN-VAE	0.444 ± 0.008	0.530 ± 0.043	0.459 ± 0.042	0.542
ODE-RNN	0.515 ± 0.095	0.573 ± 0.097	0.495 ± 0.119	0.694
L-ODE	0.592 ± 0.048	0.597 ± 0.042	0.598 ± 0.036	0.701
PrimeNet	$\mathbf{0.641 \pm 0.071}$	$\mathbf{0.663 \pm 0.047}$	$\mathbf{0.681 \pm 0.026}$	**0.842**

Figure 2.5(b). The task-specific layers typically consists of some fully connected layers with non-linear activation.

2.6 Experiments

We evaluate PrimeNet on real-world irregular and asynchronous time-series data from Healthcare and IoT domain for classification, regression, and interpolation tasks.

2.6.1 Datasets

We use the following datasets:

- **PhysioNet Challenge 2012** [106] and **MIMIC-III** [58] are multivariate time series datasets consisting of 37 and 12 physiological variables, respectively, extracted from intensive care unit (ICU) records. Each record contains 48 hours of measurements after admission to ICU. We predict in-hospital mortality (binary classification) from this data.
- **Activity** [59] dataset has 3-D positions of the waist, chest and ankles from 5 individuals

Table 2.2. Classification on MIMIC-III [58] measured by Area Under the ROC curve (AUC). Higher AUC is better.

Baselines	1-shot	2-shot	4-shot	Full
Self-supervised regular time-series methods				
TNC	0.558 ± 0.036	0.568 ± 0.053	0.555 ± 0.031	0.749
TS2Vec	0.559 ± 0.039	0.560 ± 0.028	0.565 ± 0.029	0.824
TST	0.542 ± 0.081	$\underline{0.596 \pm 0.079}$	0.591 ± 0.053	0.822
Irregular time-series methods				
mTAND	0.534 ± 0.003	0.538 ± 0.012	0.528 ± 0.003	$\underline{0.829}$
GRU-Mean	$\underline{0.573 \pm 0.010}$	0.568 ± 0.005	0.576 ± 0.007	0.786
P-LSTM	0.546 ± 0.086	0.573 ± 0.033	$\underline{0.595 \pm 0.083}$	0.745
RNN-VAE	0.516 ± 0.001	0.516 ± 0.002	0.516 ± 0.003	0.512
ODE-RNN	0.552 ± 0.005	0.562 ± 0.009	0.564 ± 0.009	0.709
L-ODE	0.481 ± 0.005	0.485 ± 0.004	0.484 ± 0.003	0.590
PrimeNet	$\mathbf{0.595 \pm 0.063}$	$\mathbf{0.601 \pm 0.06}$	$\mathbf{0.638 \pm 0.038}$	**0.838**

performing activities including walking, sitting, lying, standing, etc.

- **Appliances Energy** [115] dataset contains 138 time series with 24 dimensions, including temperature, humidity, pressure, wind speed, visibility, and dew point. The data is averaged for 10 minutes and spans 4.5 months.

PhysioNet, MIMIC-III, and Activity are naturally irregular, i.e., data was sampled at irregular times during collection. Appliances Energy is a regularly sampled dataset where we synthetically induce irregularity by dropping out random data. To better understand their irregularity pattern, we provide some summary statistics: (mean, standard deviation) of the missing ratio of each feature's time series across the dataset. If a dataset was sampled for 100 timestamps, then a 0.75 mean missing ratio means that on average, each feature was present for 25 and was missing for the remaining 75 timestamps across this dataset. Missing ratio statistics: PhysioNet $(0.86, 0.24)$, MIMIC-III $(0.65, 0.36)$, Activity $(0.75, 0.64)$, Appliances Energy $(0.87, 0.47)$.

Table 2.3. Interpolation on PhysioNet at 50% [106] ($\times 10^{-2}$), measured by RMSE. Lower RMSE is better.

Baselines	1-shot	2-shot	4-shot	Full
Self-supervised regular time-series methods				
TNC	63.33 ± 1.91	60.48 ± 3.75	47.73 ± 2.27	20.55
TS2Vec	69.83 ± 6.79	60.34 ± 2.15	45.97 ± 3.32	23.98
TST	62.99 ± 1.57	52.80 ± 3.25	50.11 ± 2.03	24.81
Irregular time-series methods				
mTAND	$\underline{62.17 \pm 8.91}$	$\underline{51.08 \pm 4.79}$	$\underline{40.35 \pm 3.91}$	$\underline{20.46}$
GRU-Mean	75.44 ± 1.13	74.41 ± 0.86	72.87 ± 0.59	30.49
P-LSTM	186.81 ± 3.45	186.61 ± 3.03	183.61 ± 2.10	31.94
RNN-VAE	194.18 ± 0.25	194.20 ± 0.46	194.05 ± 0.31	61.41
ODE-RNN	112.84 ± 7.85	107.79 ± 4.68	103.55 ± 5.34	26.69
L-ODE	97.67 ± 5.92	92.52 ± 2.53	85.96 ± 2.85	22.51
PrimeNet	$\mathbf{60.84 \pm 21.62}$	$\mathbf{41.56 \pm 16.98}$	$\mathbf{25.45 \pm 3.28}$	$\mathbf{14.3}$

2.6.2 Baselines

We compare PrimeNet with the following set of baselines:

Self-Supervised Regular Time-Series Methods

These are methods that learn from unlabeled data but assume regular time intervals.

1. **TS2Vec** [142] performs hierarchical contrastive learning over augmented context views.
2. **TNC** [118] defines temporal neighborhoods from local smoothness of data.
3. **TST** [144] pre-trains Transformer by masking fixed length segments and reconstructing them.

Irregular Time-Series Methods

These are methods that consider irregular time interval information but require a lot of labeled data to learn. They can be divided into 3 categories: (a) Recurrence-based, (b) ODE-based, and (c) attention-based fully- and semi-supervised methods.

Table 2.4. Interpolation on Activity at 10% [59] ($\times 10^{-2}$), measured by RMSE. Lower RMSE is better.

Baselines	1-shot	2-shot	4-shot	Full
Self-supervised regular time-series methods				
TNC	14.28 ± 3.07	13.92 ± 2.98	11.63 ± 2.72	7.18
TS2Vec	15.37 ± 3.71	$\underline{12.96 \pm 3.24}$	11.53 ± 2.79	7.69
TST	$\underline{14.03 \pm 3.11}$	13.58 ± 2.91	10.87 ± 2.58	6.92
Irregular time-series methods				
mTAND	15.21 ± 3.16	13.73 ± 3.02	$\underline{10.83 \pm 2.74}$	$\underline{6.89}$
GRU-Mean	18.34 ± 3.72	17.24 ± 3.61	14.82 ± 3.21	8.83
P-LSTM	42.18 ± 7.84	40.10 ± 6.81	36.53 ± 6.28	14.27
RNN-VAE	47.91 ± 8.26	41.92 ± 7.72	38.35 ± 6.89	19.37
ODE-RNN	22.71 ± 4.82	19.26 ± 4.13	18.91 ± 3.67	11.91
L-ODE	18.63 ± 3.79	16.32 ± 3.21	15.52 ± 2.62	9.26
PrimeNet	$\mathbf{11.28 \pm 2.91}$	$\mathbf{9.37 \pm 2.72}$	$\mathbf{7.85 \pm 1.98}$	**3.59**

1. **GRU-Mean** [12] combines hidden state decay with input decay.

2. **P-LSTM** [87] adds a learnable oscillator to modulate LSTM to create dependencies on elapsed-time, and uses vanishing factor in gradients.

3. **RNN-VAE** VAE model with RNN encoder and decoder.

4. **ODE-RNN** [100] uses neural ODEs to model hidden state dynamics and RNN to update hidden states with new observations.

5. **L-ODE** [100] Latent ODE with ODE-RNN encoder and neural ODE decoder.

6. **mTAND** [105] Multi-time attention module followed by a VAE-based encoder-decoder.

2.6.3 Experimental Protocols

We infer a continuous missing segment of 10% and 50% values for interpolation, while conditioning on the remaining 90% and 50% of the observed points for Activity and PhysioNet, respectively. For interpolation, we use the entire output representation from PrimeNet, while

Table 2.5. Regression on Appliances Energy [115], measured by RMSE. Lower RMSE is better.

Baselines	1-shot	2-shot	4-shot	Full
Self-supervised regular time-series methods				
TNC	12.51 ± 0.56	11.68 ± 0.51	9.82 ± 0.73	3.78
TS2Vec	3.95 ± 0.09	$\underline{3.94 \pm 0.10}$	3.73 ± 0.30	3.40
TST	3.93 ± 0.44	4.11 ± 0.62	3.96 ± 0.56	3.43
Irregular time-series methods				
mTAND	4.25 ± 1.10	4.13 ± 0.81	$\underline{3.52 \pm 0.12}$	$\underline{3.39}$
GRU-Mean	$\underline{3.91 \pm 0.46}$	3.97 ± 0.65	3.56 ± 0.15	3.44
P-LSTM	8.53 ± 0.43	7.74 ± 0.43	7.07 ± 0.30	6.58
RNN-VAE	12.51 ± 0.80	12.31 ± 0.65	11.65 ± 0.57	6.79
ODE-RNN	12.32 ± 0.90	11.29 ± 0.55	9.18 ± 0.74	6.74
L-ODE	6.82 ± 1.05	5.42 ± 1.42	4.33 ± 0.52	3.51
PrimeNet	$\mathbf{3.66 \pm 0.54}$	$\mathbf{3.64 \pm 0.43}$	$\mathbf{3.44 \pm 0.03}$	**3.21**

Table 2.6. Classification performance of PrimeNet's ablation study on PhysioNet.

Ablations	1-shot	2-shot	4-shot	Full
w/o pre	0.610 ± 0.02	0.596 ± 0.01	0.613 ± 0.01	0.829
(1)	0.624 ± 0.03	0.645 ± 0.05	0.651 ± 0.04	0.811
(2)	0.618 ± 0.06	0.638 ± 0.05	0.647 ± 0.04	0.817
(3) + (2)	0.620 ± 0.06	0.631 ± 0.07	$\underline{0.658 \pm 0.01}$	$\underline{0.834}$
(4) + (1)	$\underline{0.635 \pm 0.08}$	$\underline{0.649 \pm 0.07}$	0.655 ± 0.06	0.829
PrimeNet	$\mathbf{0.641 \pm 0.07}$	$\mathbf{0.663 \pm 0.05}$	$\mathbf{0.681 \pm 0.03}$	**0.842**

for classification and regression we use the final hidden state of `[CLS]` symbol as the aggregate sequence representation. We compute Cross-Entropy Loss for classification and Root Mean Squared Error (RMSE) for regression and interpolation. Due to class imbalance in Physionet and MIMIC-III, we assess classification using Area Under the ROC curve (AUC score). We assess interpolation and regression using RMSE.

During pretraining, we measure contrastive learning classification (i.e. how many samples are predicted correctly among the $2B$ sub-samples) and use the validation accuracy for early stopping. During finetuning, we update the parameters of both the task-specific layers and PrimeNet.

We conduct grid search on hyper-parameters, $\eta = (0.3, 0.4, 0.5, 0.6, 0.7)$, $\alpha = (0.15, 0.05, 0.03)$, $J = (1, 3, 5)$, $\mu_l, \lambda_l = (0.3, 0.4)$ and $\mu_u, \lambda_u = (0.7, 0.6)$ to report test results based on the best held-out validation performance. Best values for $\eta = 0.5, 0.6, 0.5$, 0.5 for PhysioNet, MIMIC-III, Activity, Appliances Energy, respectively.

2.6.4 Results

Table 2.1, 2.2, 2.3, 2.4, and 2.5 show the results. *k*-shot refers to *k* labeled training examples. For each few-shot setup, we repeat an experiment five times using a different training sample set each time, to report the mean and standard deviation of metrics. We mark the **best** and <u>second best</u> values.

Classification

Table 2.1 and 2.2 show AUC scores on mortality prediction task (binary classification) of PhysioNet and MIMIC-III datasets. For the fully supervised irregular time-series methods that can not leverage unlabeled data, the AUC score hovers around 0.5, suggesting that these methods do not learn any useful information from such limited labeled data, performing slightly better than a random classifier. Moreover, their performance do not improve by much when we increase the amount of labeled data.

For PhysioNet, the self-supervised regular time-series method TNC [118] performs better than irregular methods under few-shot. This may occur because the irregularity information may not be much relevant to the predictive task for PhysioNet and learning just the feature space is adequate to give good performance. Hence, methods like TNC gets significant performance gain by leveraging additional unlabeled data, despite not utilizing any time interval information. However, for MIMIC-III, the irregular time-series methods are better. For MIMIC-III, it may be crucial to learn from the irregular time gap information to do well on the predictive task. Therefore, despite using only a handful of labeled examples, irregular time-series methods can outperform the self-supervised regular time-series ones that leveraged a lot of unlabeled data.

Interpolation & Regression

Table 2.4 and 2.3 show the RMSE values of the interpolation task on Activity [59] and PhysioNet [106] datasets, while Table 2.5 shows the same for regression on Appliances Energy [116] dataset. The extremely poor performance of recurrence methods, like P-LSTM, RNN-VAE, and ODE-RNN on both tasks may be attributed to their recurrent nature that accumulates error through time, resulting in poor long-horizon imputations for time series that are sparsely observed. For both tasks, the performance of most methods improves as we increase the number of shots.

Across classification, interpolation, and regression, PrimeNet outperforms self-supervised regular time-series methods under all settings, owing to PrimeNet's ability to explicitly model for irregularity. Moreover, PrimeNet outperforms the fully and semi-supervised irregular time-series models under few-shot settings by transferring the knowledge learned from unlabeled data during pre-training to fine-tuning. Moreover, PrimeNet also outperforms the irregular time-series models under full-training data setting, when all models are trained with the entire labeled training set. This shows that the pretrain-finetune setup not only improves performance under few-shot scenarios but also when there is sufficient labeled data to train the downstream task. PrimeNet is robust to varying extents of irregularity since it excelled on datasets with widely different missing ratios.

2.6.5 Ablations

Table 2.6 shows the results of PrimeNet's ablation on PhysioNet classification. (1) shows time-sensitive contrastive learning; (2) shows constant time data masking for reconstruction; (3) shows random-sampling-based contrastive learning; (4) shows constant length data masking for reconstruction, and PrimeNet = (1) + (2). If we do not pre-train PrimeNet (first row) and directly fine-tune it on labeled data, its performance is similar to mTAND [105] (0.829 vs 0.837) as both use similar architecture. All remaining ablations were pre-trained and therefore show improved performance. (1) and (2) are pre-trained using TimeCL and TimeReco, respectively, with minimal improvement. The next two ablations pre-train using both objectives but (3) + (2) uses random sampling to generate sub-sequences, while (4) + (1) masks a constant *length* of data instead of constant *time*. Results show that neither beats PrimeNet, substantiating time-sensitive sampling and constant time masking.

2.7 Conclusion

We propose a self-supervised representation learning approach to model irregular and asynchronous multivariate time-series. We use time-sensitive contrastive learning that preserves an approximate sampling density distribution of the data to learn from representative sub-sequences. We use time-sensitive data reconstruction to mask a fixed duration of data, instead of a fixed number of points, making reconstruction tractable across regions of varying sampling density. Our pre-trained model is then fine-tuned on downstream end tasks. Experiment results show that PrimeNet outperforms both fully- and semi-supervised irregular time-series and self-supervised regular time-series methods on classification, interpolation and regression tasks across several real-world datasets. In future, we plan to apply this to irregular time-series forecasting and unsupervised anomaly detection.

Chapter 2 incorporates material from the publication “PrimeNet: Pre-training for Irregular Multivariate Time-Series”, by Ranak Roy Chowdhury, Jiacheng Li, Xiyuan Zhang, Dezhi Hong, Jingbo Shang, Rajesh K. Gupta, published in Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI 2023). The dissertation author was the primary investigator and the lead author of this paper.

Chapter 3

Task-specific Self-supervision

Building on the strong latent space for time-series established by the pre-trained model from Chapter 1, this chapter explores how to leverage this foundation for faster and more effective fine-tuning on specific tasks with less data and computational resources.

Time-series data contains temporal order information that can guide representation learning for predictive end tasks (e.g., classification, regression). Recently, there are some attempts to leverage such order information to first initialize a time-series model by reconstructing time-series values of randomly masked time segments, followed by an end-task fine-tuning on the same dataset, demonstrating improved end-task performance. However, this learning paradigm decouples data reconstruction from the end task. We argue that the representations learnt in this way are not informed by the end task and may, therefore, be sub-optimal for the end-task performance. In fact, the importance of different timestamps can vary significantly in different end tasks. We believe that representations learnt by reconstructing *important* timestamps would be a better strategy for improving end-task performance.

In this work, we propose TARNet [1], **T**ask-**A**ware **R**econstruction **Net**work, a fine-tuned model that piggybacks on the pre-trained PrimeNet backbone to learn task-aware data reconstruction that augments end-task performance. Specifically, we design a data-driven masking strategy that uses self-attention score distribution from end-task training to sample timestamps deemed important by the end task. Then, we mask out data at those timestamps

[1] Code is publicly available at https://github.com/ranakroychowdhury/TARNet

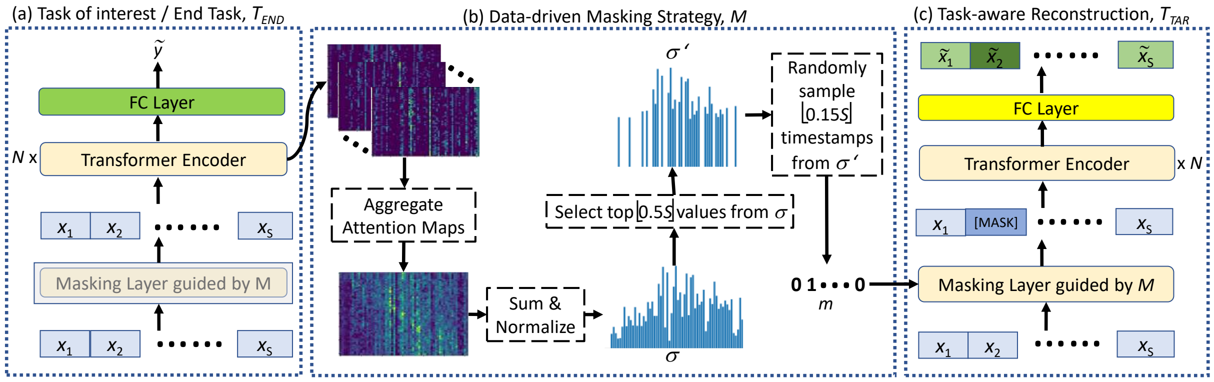

(a) Task of interest / End Task, T_{END}
$\tilde{y}$
FC Layer
N x
Transformer Encoder
x_1 x_2 x_S
Masking Layer guided by M
(b) Data-driven Masking Strategy, M
Aggregate Attention Maps
Sum & Normalize
σ
σ'
Select top $\lfloor 0.5S \rfloor$ values from σ
Randomly sample $\lfloor 0.15S \rfloor$ timestamps from σ'
0 1 • • • 0
m
(c) Task-aware Reconstruction, T_{TAR}
$\tilde{x}_1$ $\tilde{x}_2$ $\tilde{x}_S$
FC Layer
Transformer Encoder
x N
[MASK]
Masking Layer guided by M

Figure 3.1. TARNet Overview.

and reconstruct them, thereby making the reconstruction task-aware. This reconstruction task is trained alternately with the end task at every epoch, sharing parameters in a single model, allowing the representation learnt through reconstruction to improve end-task performance. Extensive experiments on tens of classification and regression datasets show that TARNet significantly outperforms state-of-the-art baseline models across all evaluation metrics.

3.1 Problem Statement

Time-series data has domain-specific structural properties encoded in the temporal ordering of events. These intrinsic properties can provide a rich source of supervision besides target labels, which the state-of-the-art time-series models [7, 148] often neglect. Recently, time-series Transformer [144] leveraged this unlabeled data to craft a reconstruction task that masks time-series values of randomly chosen time segments and reconstructs them. The pre-trained model is then fine-tuned on an end task, by reusing the *same* data samples along with their labels, leading to improved performance over exclusively doing supervised learning on the end task.

However, this data reconstruction task precedes fine-tuning as a decoupled step, which means the representation learnt during reconstruction is not informed about the end task. Hence, such learnt representation may not be fully leveraged to perform optimally on the end task.

Depending on the end task, different properties of the given data may be useful for different end tasks. For example, consider the following end tasks using the same data collected from sensors in a building: predict the level of energy consumption (high, medium, low) and the occupancy status (occupied or not) of a room based on outdoor temperature and humidity, and light intensity and CO_2 readings from a room. Energy consumption prediction task may be highly correlated to times when temperature is high (air conditioning stays on) or light intensity is high (lights are switched ON) while occupancy status may correlate to timestamps when CO_2 level is high. Hence, depending on the end task, certain timestamps in the data may be more important than others for that task.

Generic learnt representations typically result from decoupled data reconstruction and end tasks. To optimize the performance for an end task, we customize the learnt representation for the end task in TARNet. We test and validate the hypothesis that a representation learnt by reconstructing data from timestamps important to the end task will yield improved performance over reconstruction on random time segments. Therefore, we design a data reconstruction task which masks data from those important timestamps and reconstructs them. In the process, the model learns a task-specific representation, resulting in improved end task performance.

Figure 3.1 shows TARNet's learning process. Using a transformer encoder [122] as the backbone model, we train for the end task (Figure 3.1(a)) and the data reconstruction task (Figure 3.1(c)) alternately on the same model. In order to compute the timestamps to mask during data reconstruction, we design a data-driven masking strategy (Figure 3.1(b)). It uses the self-attention score distribution generated by transformer encoder during the end task training and determines the set of timestamps to mask. Since the two tasks share parameters, the representation learnt during reconstruction can be effectively leveraged by the end task to improve performance.

We conducted experiments on 34 classification datasets from UEA ARCHIVE [6], UCI MACHINE LEARNING REPOSITORY [34, 49] and 6 regression datasets from MONASH UNIVERSITY, UEA, UCR TIME SERIES REGRESSION ARCHIVE [114]. Time Series Transformer (TST) [144], the current state-of-the-art for time-series, achieved the best accuracy on 6 out of 10 datasets, when compared with 5 baselines. We compared TARNet with 14 state-of-the-art baselines and it performed the best on 17 out of 34 datasets, being 2.7% higher in average accuracy than TST, which now performs best on 7 datasets. Similarly, TST achieved the lowest error on 3 out of 6 datasets for regression when compared with 11 state-of-the-art baselines. TARNet achieved the lowest error on 3 and 2^{nd} lowest error on 2 datasets when compared with the same baselines, whereas TST now achieves the lowest error on 2 and 2^{nd} lowest error on 1 dataset. We conducted case studies to show how TARNet's data-driven masking strategy learns task-specific representations, consistent with domain characteristics, thereby boosting end-task performance.

In summary, our main contributions are:

- We propose TARNet to learn task-aware reconstruction from time-series data to augment end-task performance.
- We design a data-driven masking strategy to determine important timestamps to an end task and learn to reconstruct them.
- We evaluate TARNet on numerous real-world datasets to validate and quantify its efficacy compared with state-of-the-art methods.

3.2 Related Work

3.2.1 Non-Deep Learning Methods

ROCKET [27] and MiniROCKET [28] recently produced state-of-the art results for time-series. They learn features extracted by numerous and various random convolutional kernels. Other relevant directions include: (1) time series shapelet, (2) bag-of-patterns, and (3) distance-based models. Baydogan [9] introduced Symbolic Representation to learn local relationships between different dimensions. Shapelets [139] are short discriminative time series sub-sequences, e.g. dynamic shapelets [79], efficient shapelets [51]. WEASEL-MUSE [101] utilizes bag of SFA (Symbolic Fourier Approximation). Distance-based methods [134, 31] use distance metric to measure similarity of a pair of time series. Among limitations of these approaches are that they incorporate expert insights, consist of large, heterogeneous ensembles of classifiers, scale poorly to long time-series, and many apply to only uni-variate time-series.

TARNet can be applied to both uni- and multi-variate time-series, automatically extracts features, and handles long time-series.

3.2.2 Deep Learning Methods

Using labeled data

Fawaz [36] summarize many neural networks-based methods for time-series. Most neural networks-based methods use some arrangement of LSTM, CNN or both [150, 60]. Others use different components of neural models, e.g., learnable temporal pooling [70], correlative channel-aware learnable fusion [7], label-learning [78], attentional prototype network [148], and shapelet embedding [71]. TARNet proposes a subsidiary data reconstruction technique that utilizes knowledge from the end task to learn a task-specific data representation. Sharing parameters of this reconstruction task with the end task in a single architecture allows the learnt representation to improve end task performance.

Using both unlabeled and labeled data

Unsupervised representation learning for time-series uses triplet loss with negative sampling [38], hierarchical contrastive loss [143], temporal and contextual contrasting [35], local smoothness to define neighborhoods in time [118], and reprogramming acoustic models [137]. TST [144] first pre-trains a transformer model by an unsupervised objective; masks out time-series values at random time segments from data and reconstructs them. It then reuses the *same* training samples to fine-tune the model on an end task. This gave improved performance than using the data once to train a fully supervised model.

However, decoupling the data reconstruction from the end task makes the representation learnt during reconstruction uninformed about the end task. Depending on the end task, certain timestamps in time-series data may be more important than others [74], which the learnt representation ignores. TARNet aims to learn a task-aware data reconstruction by masking important timestamps with respect to the end task. Hence, the learnt representation is better suited for improving end task performance than the representation learnt from reconstructing randomly masked time segments.

3.3 TARNet Framework

In Figure 3.1, we show a schematic diagram of TARNet common across all considered tasks. In this section, we first present the problem setting and base model architecture shared by the two tasks. Then, we explain the end-task T_{END} (i.e., Figure 3.1(a)) and task-aware reconstruction T_{TAR} (i.e, Figure 3.1(c)). Finally, we present our data-driven masking strategy (i.e., Figure 3.1(b)) that uses information from T_{END} to decide which timestamps to mask for T_{TAR}.

3.3.1 Problem Description and Notations

Each training sample $\mathrm{X} \in \mathbb{R}^{S \times N}$ denotes a multivariate time-series of length S and N variables. Specifically, it comprises a sequence of S N-dimensional feature vectors, $x_t \in \mathbb{R}^N : \mathrm{X} \in \mathbb{R}^{S \times N}$. This formulation also covers the uni-variate case when $N = 1$. All the training samples come together with a target label y, which is an integer class id for a classification task or a real-valued number for a regression task. The full training dataset is labeled, i.e. we do not leverage any additional unlabeled data. Based on these training samples, we build a model to predict the label $\tilde{y}$ of unseen data X.

3.3.2 Base Model

We opt to use Transformer Encoders [122] as the backbone model, as we aim to develop a general framework to learn task-specific reconstruction that can be applied for a multitude of tasks. An architecture consisting of an encoder provides flexibility as it can not only handle tasks like classification, regression, imputation, but also handle generative tasks such as forecasting. One can plug in a task of interest by replacing the Fully Connected (FC) Layer in Figure 3.1(a) by task-specific layers (e.g., decoder for forecasting).

The feature vectors x_t are first mean-standardized per variable dimension. Then x_t is linearly projected onto a D-dimensional vector space, where D is the dimension of the Transformer model sequence element representations (typically called embedding dimension):

$$u_t = \mathrm{W}_p x_t + b_p, \tag{3.1}$$

where $\mathrm{W}_p \in \mathbb{R}^{D \times N}$, $b_p \in \mathbb{R}^D$ are learnable parameters and $u_t \in \mathbb{R}^D, t = 1, 2, ..., S$ are the model input vectors. The Transformer is a feed-forward architecture insensitive to the ordering of input. Therefore, we add positional encoding to these input vectors in order to make it aware of the sequential nature of the time series. The resultant vectors become the queries, keys and values of the self-attention layer in the encoder block. We pass data through several layers of such Transformer encoder blocks. Then, we pass the output values weighted by self-attention scores through a fully connected feed-forward network. We refer the reader to the original work [122] for a detailed description of the Transformer model.

3.3.3 End Task (T_{END})

For clarity, we use classification and regression as example end tasks here. Please note that TARNet can be easily extended to other tasks such as anomaly detection and time-series forecasting, by tweaking the FC Layer in Figure 3.1(a).

We modify the base model architecture presented in Section 3.3.2 for regression and classification in the following way:

The data fed to T_{END} is not masked, as illustrated by the frozen Masking Layer in Figure 3.1(a). The vector corresponding to the last timestamp from Transformer Encoders $z_t \in \mathbb{R}^D$ is fed through 2 FC layers and $RELU$ activation (represented as f), with parameters

$$\mathrm{W}_{L1} \in \mathbb{R}^{K_E \times D}, b_{L1} \in \mathbb{R}^{K_E}, \mathrm{W}_{L2} \in \mathbb{R}^{K_E \times K_E}, b_{L2} \in \mathbb{R}^{K_E},$$

followed by the output layer with parameters

$$\mathrm{W}_E^O \in \mathbb{R}^{C \times K_E}, b_E^O \in \mathbb{R}^C,$$

where K_E is the feed-forward dimension of FC Layer for T_{END} and C is the number of classes for classification or number of scalars to be estimated for regression (typically $C = 1$):

$$\tilde{y} = \mathrm{W}_E^O f(\mathrm{W}_{L2} f(\mathrm{W}_{L1} z_t + b_{L1}) + b_{L2}) + b_E^O. \tag{3.2}$$

For classification, predictions $\tilde{y}$ are passed through a softmax to give probability distribution, p, over C classes. We use cross-entropy loss with categorical ground truth labels, $L_{END} = \sum_{i=1}^{C} y_i log(p_i)$. For regression, we use squared error, $L_{END} = \|\tilde{y} - y\|_2^2$.

3.3.4 Task-aware Reconstruction (T_{TAR})

Learning data representation through reconstruction has been explored in natural language processing [29] and time-series [144]. The goal of T_{TAR}, illustrated in Figure 3.1(c), is to learn a data representation by reconstructing the input data X after it has been appropriately masked by the Data-driven Masking Strategy, M.

The role of TARNet's masking strategy M, elaborated in Section 3.3.5, is to generate a new binary training data mask $m \in \mathbb{R}^S$ for each training sample at every epoch. It is a boolean array with $\lfloor \mu S \rfloor$ number of 1's, where μ is a hyper-parameter $0 < \mu < 1$, to select the timestamps to be masked from X for the reconstruction task. Let m_t represent the value of m at timestamp t. If $m_t = 1$ we mask x_t, otherwise we do not. Masking a particular timestamp, t, involves replacing the N-dimensional feature vector x_t with zeros. X passes through Transformer Encoder layers after being masked by m. The final representation vectors $\mathrm{Z} \in \mathbb{R}^{S \times D}$ is fed through 2 FC layers and $RELU$ activation, with parameters

$$\mathrm{W}_{L3} \in \mathbb{R}^{K_R \times D}, b_{L3} \in \mathbb{R}^{K_R}, \mathrm{W}_{L4} \in \mathbb{R}^{K_R \times K_R}, b_{L4} \in \mathbb{R}^{K_R},$$

followed by the output layer with parameters

$$\mathrm{W}_R^O \in \mathbb{R}^{N \times K_R}, b_R^O \in \mathbb{R}^N,$$

where K_R is the feed-forward dimension of FC Layer for T_{TAR} and N is the number of variables:

$$\tilde{\mathrm{X}} = \mathrm{W}_R^O f(\mathrm{W}_{L4} f(\mathrm{W}_{L3} Z + b_{L3}) + b_{L4}) + b_R^O. \tag{3.3}$$

The label for this task is the raw input data X. To ensure accurate reconstruction, we calculate Mean Square Error (MSE) between the ground truth X and prediction $\tilde{\mathrm{X}}$. We calculate the average MSE loss for masked and unmasked part of the data as follows:

$$L_{masked} = \frac{1}{N \sum_{t=1}^{S} m_t} \sum_{t=1}^{S} m_t \|\tilde{\mathrm{x}}_t - \mathrm{x}_t\|_2^2, \tag{3.4}$$

$$L_{unmasked} = \frac{1}{N(S - \sum_{t=1}^{S} m_t)} \sum_{t=1}^{S} (1 - m_t) \|\tilde{\mathrm{x}}_t - \mathrm{x}_t\|_2^2. \tag{3.5}$$

Unlike TST, which only considers MSE loss for reconstructing the masked portion of the data, L_{masked}, we include loss incurred for replicating the unmasked, observed portion of the input data, $L_{unmasked}$, as well. Time-series data is auto-regressive with strong correlation across time. Therefore, the ability to reconstruct the masked data at a given timestamp depends on how effectively the model learns to reconstruct the unmasked data and use that as context to infer the masked data. Including the loss for the unmasked data ensures its accurate reconstruction.

The combined reconstruction loss L_{TAR} is a weighted sum of L_{masked} and $L_{unmasked}$, given by

$$L_{TAR} = \lambda L_{masked} + (1 - \lambda) L_{unmasked}, \tag{3.6}$$

where λ is a hyper-parameter $0 < \lambda < 1$ that controls the relative weights between the two losses. It is advisable to keep $\lambda > 0.5$ because the masked timestamps are more important for the end task than the unmasked ones.

With L_{END} as the end task loss, the total loss becomes

$$L_{Total} = \eta L_{TAR} + (1 - \eta) L_{END}, \tag{3.7}$$

where η is a hyper-parameter ($0 < \eta < 1$) that controls the relative weights between the two task losses. We train T_{END} and T_{TAR} end-to-end alternately at every epoch, until convergence.

Algorithm 3. Training of TARNet

Input: X, y
Hyper-parameters: μ, β, λ, η
Output: *Model*

```
1:  $\sigma$ initialized randomly
2:  $Model$ = TransformerEncoder()
3:  while training do
4:      $\sigma' =$ top $\lfloor \beta S \rfloor$ values from $\sigma$
5:      $m \sim$ Randomly sample $\lfloor \mu S \rfloor$ timestamps without replacement from $\sigma'$
6:      $\tilde{\mathrm{X}}, \tilde{y}, \mathrm{A} = Model$.train(X, $m$) # A $\leftarrow$ Self-Attention Scores
7:      Compute $L_{TAR}(\tilde{\mathrm{X}}, \mathrm{X}, \lambda)$ and $L_{END}(\tilde{y}, y)$
8:      $L_{Total} = \eta L_{TAR} + (1 - \eta) L_{END}$
9:      $\sigma =$ add_and_normalize(A)
10: end while
11: return $Model$
```

3.3.5 Data-driven Masking Strategy (M)

Data reconstruction in Time-series Transformer [144] involves masking segment of time-series data at randomly chosen timestamps and reconstructing them. However, different timestamps in the data may have different levels of importance to the end task. Therefore, we eschew random reconstruction of data in favor of a strategy that uses end task characteristics. Specifically, we identify timestamps that the end task deemed *important* during learning. We will then mask x_t from X corresponding to those timestamps and reconstruct them during T_{TAR}. We hypothesize that reconstructing data at timestamps identified to be important by the end task will generate a data representation that benefits the end task. This is in contrast to a random masking based data reconstruction, which does not consider any such information.

To define the notion of an "important" timestamp, we use self-attention weights generated by Transformer Encoder in the forward pass of T_{END}. Attention weights indicate how much weight should be assigned to each x_t to compute representation for a given x_t. We compute aggregate attention map $\mathrm{A} \in \mathbb{R}^{S \times S}$ by summing the attention maps generated by each layer of Transformer Encoder. Let A_{ik} be the attention weight assigned to x_k during update of x_i, where

$i = k = 1, 2, ..., S$, and $\sum_{k=1}^{S} \mathrm{A}_{ik} = 1$ for all i. Therefore, the update to x_i is a weighted sum of $x_{1,2,...,S}$, where the weights are $\mathrm{A}_{i,k=1,2,...S}$. We compute $\sigma \in \mathbb{R}^S$, where $\sigma_k = \frac{\sum_{i=1}^{S} \mathrm{A}_{ik}}{\sum_{k=1}^{S} \sum_{i=1}^{S} \mathrm{A}_{ik}}$ for $k = 1, 2, ..., S$. σ_k represents the normalized aggregate attention weight of timestamp k to the computation of x_1, x_2, ..., x_S. We define the importance of each timestamp by its magnitude in σ, i.e. the higher σ_k is, the more important timestamp k is for T_{END}.

We then select the timestamps corresponding to the top $\lfloor \mu S \rfloor$ values in σ and mask them from X for reconstruction. Since the same training data is fed at every epoch, the set of important timestamps computed from a given sample will not vary across epochs. Hence, the model may memorize reconstructing a few selected timestamps from the sample, leading to *overfitting*. Considering the heterogeneity in time-series data due to irregular sampling frequency or uncertainty about feature availability, it is probable that real-world data may have a different set of important timestamps compared to those seen in training data. Therefore, not exploring enough timestamps to approximate the training data distribution may lead to poor generalization on the real-world data.

Hence, we ensure that for every sample, at each epoch the model explores a random set of timestamps among those that are important. Therefore, we introduce an attention regularization parameter, β, where $\beta > \mu$ and $0 < \beta < 1$. We, therefore, compute set σ' to choose the top $\lfloor \beta S \rfloor$ values in σ. Then we randomly sample $\lfloor \mu S \rfloor$ timestamps without replacement from σ' to generate the training data mask m. $m_t = 1$ if t is sampled from σ', otherwise 0.

Although we still choose an important set of timestamps to mask, the use of randomization through sampling ensures that the model does not always mask the same set of timestamps for a sample throughout its entire training regime. This gives the model a more versatile representation of the underlying data distribution, yet, one that is important for the end task. This data-driven masking strategy makes the model learn task-specific data representation by reconstructing data at those timestamps deemed important by the end task. Algorithm 3 outlines the training procedure of TARNet.

Example 3 Suppose the data has a sequence of length 6 and the attention scores look like following,

$$\begin{bmatrix} 0.0443 & 0.0570 & 0.3744 & 0.2279 & 0.0753 & 0.2211 \\ 0.2008 & 0.2622 & 0.1786 & 0.0145 & 0.1545 & 0.1894 \\ 0.0748 & 0.1122 & 0.0861 & 0.1611 & 0.2570 & 0.3087 \\ 0.2221 & 0.0098 & 0.2580 & 0.2412 & 0.1580 & 0.1109 \\ 0.1917 & 0.0353 & 0.2559 & 0.0305 & 0.2657 & 0.2210 \\ 0.1912 & 0.1578 & 0.2118 & 0.1020 & 0.2582 & 0.0789 \end{bmatrix}$$

Each row represents how much weight was put into the representation of each timestamp while computing the representation of a given timestamp. For example, the first row shows that while computing the resulting representation of the first timestamp, the self-attention method put 4.43% weight to the representation of the first timestamp, 5.7% to the representation of the second timestamp, 37.44% weight to the representation of the third timestamp, and so on.
If we aggregate the sum across the columns of the attention score matrix, it gives,
$\sigma = [0.9249, 0.6344, 1.3649, 0.7773, 1.1687, 1.1298]$
σ shows how much total weight was assigned to a particular timestamp for the computation of representation of all timestamps. If $\beta = 0.33$, then we will mask out the $0.33 \times 6 = 2$ most important timestamps from the data and reconstruct them. Hence, we should mask out the features from timestamp 3 and 5 because they had the highest aggregate values in σ. However, as we stated earlier, this may lead to over-fitting as the same set of 2 timestamps may be masked out at every epoch of training.
Instead, as outlined in Algorithm 3, let's consider $\beta = 0.5$ and $\mu = 0.67$, then, σ' will have a length of $0.5 \times 6 = 3$,
$\sigma' = [1.3649, 1.1687, 1.1298]$
Then, we randomly sample 0.67×3 timestamps from σ' and mask out the features corresponding to those timestamps for reconstruction. Suppose at Epoch 1, we mask out timestamps 2 and 4, while at Epoch 2 we mask out timestamps 4 and 5. Due to this randomization, we are not choosing the same set of timestamps at every epoch of training. This enables the model to generalize to new set of important timestamps that may arise in the test set. Although we are selecting these timestamps randomly, they are being sampled from σ', which is already a curated set of important timestamps from σ.

3.4 Experiments

We present the datasets, baselines, training settings, followed by the evaluation metrics. We then show and analyze classification and regression results of TARNet. We also conduct an ablation study, few-shot training experiments and case studies to justify TARNet.

3.4.1 Experimental Setup

We use benchmark time-series datasets with detailed information available in UEA ARCHIVE [6], UCI MACHINE LEARNING REPOSITORY [34, 49], and MONASH UNIVERSITY, UEA, UCR TIME SERIES REGRESSION ARCHIVE [114]. These datasets represent an assortment of domains (Motion, Audio, EEG, HAR), sensor type, and sampling frequency. The number of training data points varies from 15 to over one million, the length of the time series, S, varies between 8 to $17,984$, the number of features, N, varies between 1 to $1,345$, and the number of target classes, C, varies between 2 to 39. $N = 1$ covers the uni-variate case. $N > 1$ refers to the multi-variate case.

We compare TARNet with statistical [6, 103, 32, 112, 14, 101, 27, 28] and deep learning [60, 148, 71, 128, 36, 37, 38, 144, 142, 118, 35] baselines.

Statistical Baselines

Statistical methods for time-series analysis typically rely on assumptions about the data's underlying structure and are effective for simpler, linear patterns, such as trends and seasonality.

1. **Distance-based method** [6]. Euclidean Distance (ED), dimension independent dynamic time warping (DTWI), and dimension-dependent dynamic time warping (DTWD) [103].

2. **SVR**: [32] Support Vector Regression.

3. **Tree-based methods**: Random Forest [112] and XGBoost [14].

4. **WEASEL-MUSE** [101] is a bag-of-pattern based sliding-window approach with statistical feature extraction and filtration.

5. **Rocket** [27] convolves time series with random convolutional kernels and applies global max pooling to extract features.

6. **MiniRocket** [28] upgrades Rocket by speeding it up, using a small, fixed set of kernels, and is almost entirely deterministic.

Deep Learning Baselines

Deep learning methods can automatically learn complex, non-linear patterns from large datasets, making them more powerful for handling intricate time-series data with high variability and multiple influencing factors.

1. **FCN** [128] Fully Convolutional Networks. Replaces traditional final FC layer with a Global Average Pooling (GAP) layer.

2. **MLSTM-FCNs** [60] expands LSTM-FCN and Attention LSTM-FCN by adding squeeze-and-excitation blocks.

3. **Negative samples (NS)** [38] generates negative samples and trains a dilated causal convolution encoder with triplet loss.

4. **TapNet** [148] designs random group permutation method with multi-layer convolutional and attentional prototype network.

5. **ShapeNet** [71] extends shapelet [139] for multivariate time-series. Learns shared embedding space across different shapelet candidates, trains a dilated causal CNN, followed by an SVM.

6. **Time Series Transformer (TST)** [144] pre-trains Transformer Encoder by masking random time segments and reconstructing them. Reuses the same data to fine-tune the model.

7. **TS2Vec** [142] performs hierarchical contrastive learning over augmented context views. Builds representation of an arbitrary sub-sequence by aggregating representations of timestamps.

8. **TNC** [118] leverages local smoothness of a signal to define temporal neighborhoods and learns generalizable representations.

9. **TS-TCC** [35] encourages consistency of different data augmentations to learn transformation invariant representations.

10. **ResNet** [36] uses convolutional followed by a GAP layer. Adds shortcut residual connection between convolutional layers.

11. **Inception** [37] is an ensemble of deep CNN models, inspired by the Inception-v4 architecture.

We normalize the datasets for each of our experiments. For datasets on which the accuracies of the baselines have been reported, we present the same results according to their papers. For the remaining datasets, we train all the baseline models with sufficient hyper-parameter tuning to produce results. Since our benchmark datasets are widely heterogeneous in terms of number of data points, features, sequence length, and sampling frequency, as well as the physical nature of the data itself, we obtain better performance via cursory tuning of architecture-specific hyper-parameters. To select hyper-parameters, we do a random 80%-20% split of the training set and used the 20% as a validation set for hyper-parameter tuning. After fixing the hyper-parameters, we train the model again using the entire training set and save the model with the lowest training loss. We use the saved model to evaluate on the official test set and report our evaluation metrics.

Appendix B.0.2 explains the hardware and framework details of our experiments. Tables B.2 and B.3 in Appendix B.0.3 shows the best hyper-parameter combination for each dataset.

3.4.2 Evaluation Metrics

We use accuracy and Root Mean Squared Error (RMSE) error as our performance metric for classification and regression, respectively. Considering the large number of datasets and baselines used, it is highly unlikely for a single model to outperform all other methods on every datasets. Therefore, we also present some summary statistics to present a holistic and a fairer comparison of the methods. The evaluation metrics are as follows:

- **Ours 1-to-1 Wins/Draws/Losses**: Number of datasets for which TARNet's accuracy or RMSE is better/same/worse than the corresponding baselines, respectively. Higher wins, lower draws and lower losses are better. This is useful to draw a one-on-one comparison between TARNet and a given model.
- **Mean Rank**: Average rank of a model across all datasets. Lowest rank is assigned to model with highest accuracy for classification and lowest RMSE for regression. Lower mean rank is better.
- **Avg.Rel.Diff.Mean** [144]: We report the "average relative difference from mean" metric r_j for each model j, over N datasets:

$$r_j = \frac{1}{N}\sum_{i=1}^{N}\frac{R(i,j)-\bar{R}_i}{\bar{R}_i}, \qquad \bar{R}_i = \frac{1}{M}\sum_{j=1}^{M}R(i,j), \tag{3.8}$$

  where $R(i,j)$ is the RMSE of model j on dataset i and M is the number of models. $r_j = -0.3$ means that the model on average attains 30% lower RMSE on a dataset than the average model performance on the same dataset. Lower value is better.

3.4.3 Classification

Table 3.1, 3.2, and 3.3 shows the accuracy of TARNet and baselines on classification datasets from UEA ARCHIVE and UCI MACHINE LEARNING REPOSITORY. We mark the **best** and <u>second best</u> values. Baselines are presented in ascending order (left to right) by average accuracy. A dash indicates that the corresponding method failed to run on this dataset. Higher Total best accuracy, average accuracy, and Ours 1-to-1 Wins is better. Lower Ours 1-to-1 Draws, Ours 1-to-1 Losses, and Mean Rank is better.

According to the results, the overall accuracy of TARNet is the best among all compared methods. TARNet performs the best on 17 datasets, as compared to 7 and 6 by the next best baselines TST [144] and Rocket [27], respectively. TARNet achieves a 2.7-point higher average accuracy across all datasets over TST. The closest competitors of TARNet are TST and Rocket, but TARNet still outperforms them on 20 datasets while losing on 14 and 12, respectively. TARNet ranks 1$^{\text{st}}$ (lowest "Mean Rank") on average, having a 0.71-point lower average than the 2$^{\text{nd}}$ best MiniRocket. Rocket and ShapeNet ranks 3$^{\text{rd}}$ and 4$^{\text{th}}$ with a 1.18 and 1.47-point higher average, respectively, than TARNet.

The large number of datasets and baselines used makes it highly unlikely for a single model to outperform all other methods on every dataset. For example, TST had the 2$^{\text{nd}}$ best "Total best Accuracy" (7) and "Average Accuracy" (0.745), but it ranks 5$^{\text{th}}$ across all models, with a 1.74-point higher average than TARNet. This means that for the datasets where TST under-performs, its performance metrics are significantly below those of other baselines, pushing down its "Mean Rank." However, TARNet performs well across all evaluation metrics. Not only does it have the highest "Total best Accuracy" (17) and "Average Accuracy" (0.772), but it also ranks 1$^{\text{st}}$, meaning that for the datasets where TARNet under-performs, it still generates better performance than most of its baselines, pushing up its "Mean Rank". Moreover, we find that on datasets where TARNet under-performs, the winning methods are in fact different. Considering that no single baseline is consistently better than TARNet, as illustrated by the baselines' low number of best accuracies, low average accuracies and high mean rank, we argue that TARNet is the new benchmark for time-series classification.

Moreover, TARNet achieves the best accuracy across a diverse set of data characteristics. For example, TARNet has the best accuracy for Atrial Fibrillation and Occupancy with 15 and $1.2m+$ training data points, respectively, for RacketSports and Cricket with sequence length of 30 and 1197, respectively, for Epilepsy and FaceDetection with 3 and 44 features, respectively and for MotorImagery and OpportunityGestures with 2 and 17 classes, respectively.

3.4.4 Regression

We compare regression results against all the baselines reported by TST [144]. Table 3.4, 3.5, and 3.6 show the Root Mean Squared Error of TARNet and baselines on regression datasets from MONASH UNIVERSITY, UEA, UCR TIME SERIES REGRESSION ARCHIVE [114]. We mark the **best** and <u>second best</u> values. Baselines are presented in descending order (left to right) by mean rank. Avg.Rel.Diff.Mean: Average Relative Difference from Mean over all models, e.g. -0.3 means that the model on average attains 30% lower RMSE than the average model performance. Higher Total best loss and Ours 1-to-1 Wins is better. Lower Ours 1-to-1 Draws, Ours 1-to-1 Losses, Mean Rank, and Avg.Rel.Diff.Mean is better.

TARNet ranks 1^{st} on three and 2^{nd} on two datasets, which is better than what any of the baseline models achieve. For the overall rank, TARNet achieves an average rank of 1.833, setting it clearly apart from all other models; the overall second best model, TST [144] has an average rank of 2.5; XGB, Inception, and FCN (which outperformed TARNet on one dataset) on average ranks 4.333, 6.5, and 7, respectively. Both TST [144] and TARNet use a similar transformer backbone model which explains the small difference in Avg.Rel.Diff.Mean scores. However, TARNet still outperforms TST and all other baseline models by attaining 31.3% lower RMSE on average than the mean RMSE among all models. Considering that TARNet achieves the highest number of best losses, lowest mean rank, and lowest Avg.Rel.Diff.Mean, we argue that TARNet is the new benchmark for time-series regression.

Although TST [144] pretrains and finetunes on the same dataset, the data reconstruction and the supervised end-task runs *sequentially*, slowing down training time. However, TARNet trains both tasks, T_{TAR} and T_{END} *parallely*. Hence, not only TARNet outperforms TST on the end-task but it also trains faster than TST.

3.4.5 Ablation Study

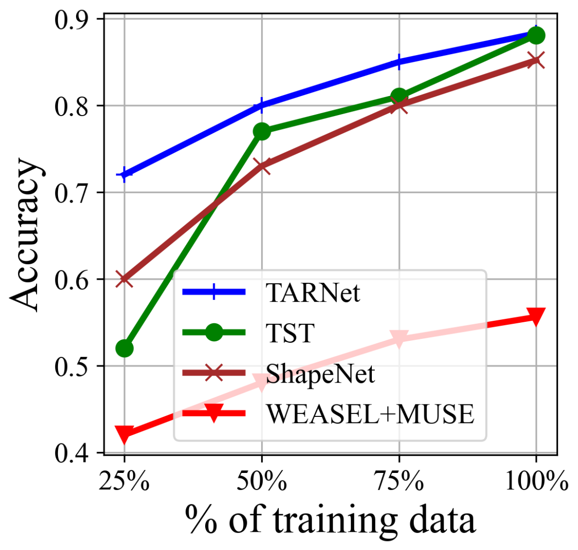
We justify our design choices of M through ablation study results on classification and regression tasks in Table 3.7. TARNet-Random uses the same architecture as TARNet but instead masks timestamps randomly and reconstructs them, giving substandard performance. TARNet-Top μ selects timestamps corresponding to the top $\lfloor \mu S \rfloor$ values in σ and masks them from X for reconstruction. This does not lead to a clear improvement which may be attributed to *overfitting*, as explained in Section 3.3.5. This prompts sampling to TARNet-Top μ while selecting the timestamps to mask from the set of important timestamps, resulting in TARNet. To ensure a fair comparison, we maintain the same set of hyper-parameters across all ablation models for each dataset. Table 3.7 shows that TARNet has the highest average accuracy, most number of datasets with highest accuracy and lowest loss, and lowest mean rank. TARNet combines ideas from both TARNet-Random and TARNet-Top μ to counter their individual drawbacks and yields better performance.

3.4.6 Can T_{TAR} compensate for limited labeled training data?

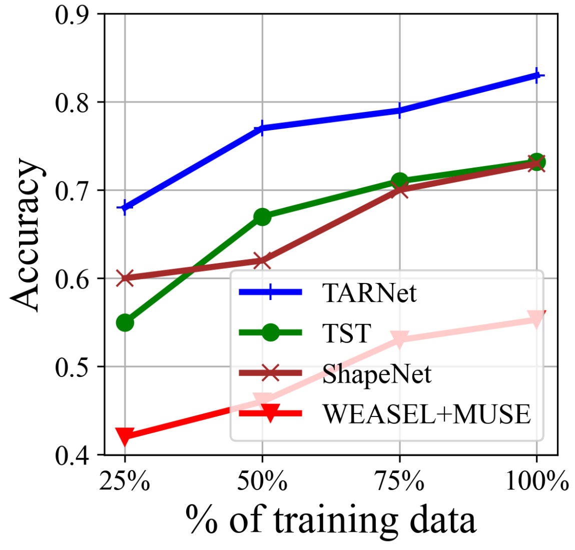
We study whether under data-deficient environments TARNet can make better use of limited data compared to baselines. This will illustrate if the knowledge gained during reconstruction, T_{TAR}, can compensate for a lack of labeled data to train the end task, T_{END}.

We choose occupancy and human gestures datasets for classification. As Figure 3.2 (a) and (b) show, the accuracy of all models increases as the amount of training data increases. Particularly, TARNet has a steep rise for both datasets, signifying that the greatest improvement occurs with low quantity of training data. Similarly, we choose LiveFuelMoisture and IEEEPPG datasets for regression. As Figure 3.2 (c) and (d) show, the RMSE Loss of all models decreases as the amount of training data increases. Even with just 25% training data, TARNet achieves significantly lower loss than any baselines. It achieves superior performance over all baselines at all quantities of training data, for both classification and regression.

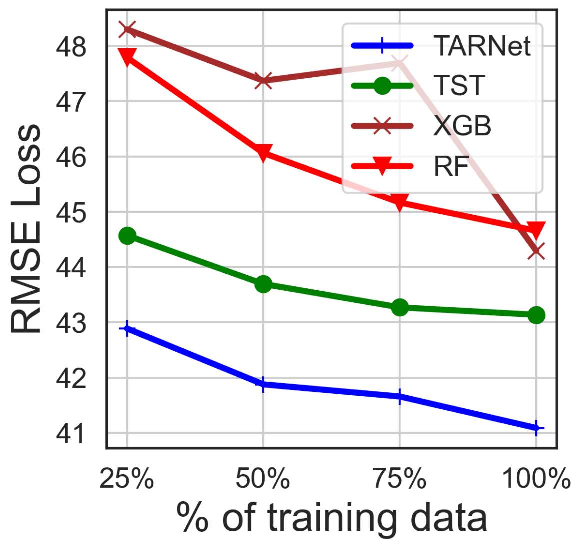
Both TST and TARNet can leverage additional information learnt though reconstruction to compensate for the lack of labeled data, resulting in better performance over other baselines. However, making the reconstruction task-aware improves the performance of TARNet over TST. For example, in Occupancy, TARNet achieves the same performance with 50% training data, which TST and ShapeNet require 75% training data to achieve. Similarly, for LiveFuelMoisture and IEEEPPG, TARNet achieves lower RMSE with just 25% and 50% training data, respectively, than TST does with 100% training data.



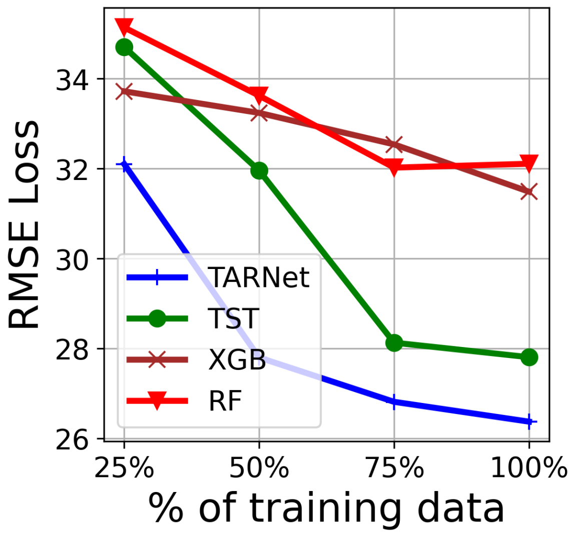
(a) Occupancy



(b) Opportunity Gestures



(c) LiveFuelMoisture



(d) IEEEPPG

Figure 3.2. (a) and (b) show classification accuracy, and (c) and (d) show regression RMSE Loss against % of training data.

3.4.7 Explaining Masking Strategy, M

We provide two real-world case studies to show why a task-aware reconstruction learnt through a data-driven masking strategy, M, is superior to a reconstruction learnt through random masking. For qualitative analysis, we show normalized aggregate attention, σ, computed from attention maps of Transformer during T_{END}.

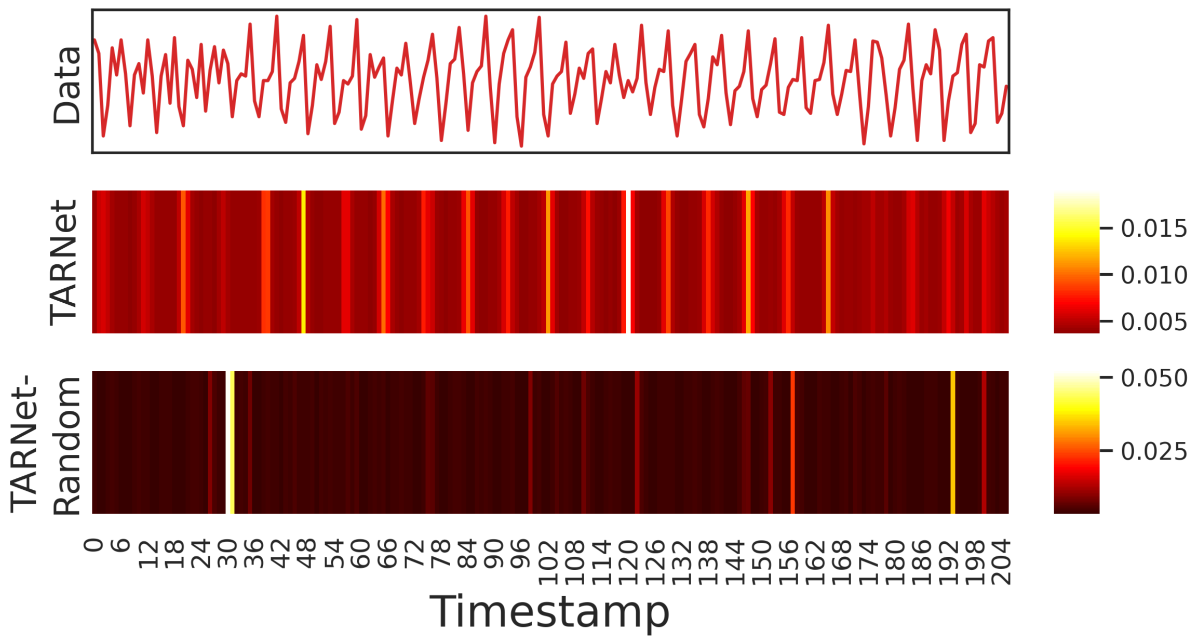


Figure 3.3. σ plotted as heatmap for Epilepsy.

Case Study I: Epilepsy Figure 3.3 shows a time-series plot of an accelerometer data from a person conducting the activity of "Sawing" (classification label). Following the time-series plot are the σ scores, as discovered by TARNet and TARNet-Random. Sawing involves strong periodic motion of the hand as illustrated by the time-series plot. Figure 3.3 shows that a random-masking based auto-regressive task (TARNet-Random) could not capture this inherent periodicity in the data, which TARNet could successfully decipher. Therefore, the accuracy achieved by TARNet and TARNet-Random is 1 and 0.75, respectively. Being able to selectively mask "important" timestamps during reconstruction in a data-driven manner enables TARNet to effectively capture the domain-specific properties from the data, leading to better classification performance.

Case Study II: Face Detection A person is shown a face image or a scrambled image and her MEG readings are recorded. The task is to determine what the person saw (classification) based on the collected MEG data. The MEG recording (response) is collected over 1.5-second but the image (stimulus) is only shown 0.5-seconds after the MEG has started recording. Figure 3.4 shows the time-series plot of a sample MEG data. Since the entire 1.5-second corresponds to 62

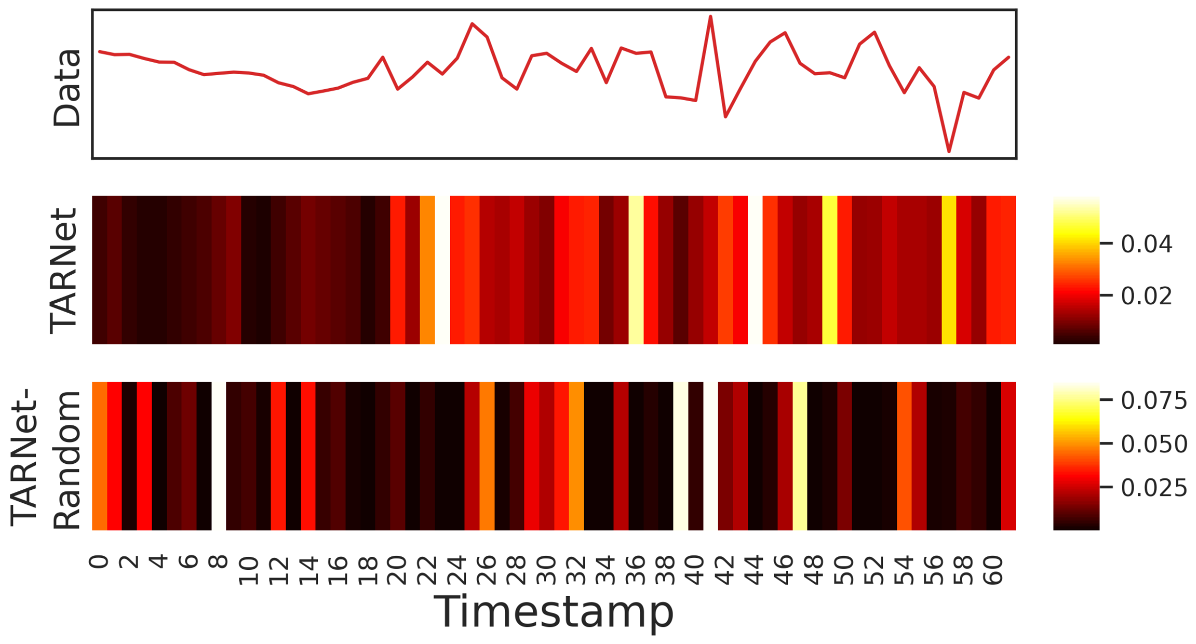


Figure 3.4. σ plotted as heatmap for Face Detection.

timestamps, this means that no stimulus was provided to the subject for the first 20 timestamps (0.5-seconds). So the discriminatory MEG response, important for classification, is received from 20-th timestamp onward, as illustrated by the onset of sudden fluctuation in signal strength. Figure 3.4 shows that TARNet assigns high σ values around the 20-th timestamp and can clearly infer the signal arrival time from the MEG response. TARNet discriminates between the "unimportant" and "important" timestamps for classification by assigning higher average attention per timestamp for times greater than 20 than to those before 20. However, TARNet-Random fails to infer such task-specific domain properties from the data and assigns attention weights randomly across time. Hence, TARNet-Random achieves an accuracy of 0.607, whereas TARNet achieves 0.641.

The two case studies substantiate why using M to decide which timestamps to mask during reconstruction is important. Representations learnt through reconstructing "important" timestamps reflect some domain-specific inherent properties in the data, as illustrated by how the attention scores have been assigned. Such domain properties are relevant to the end task and can clearly lead to performance improvement on the end task, as illustrated in Tables 3.1, 3.2, 3.3, 3.4, 3.5, and 3.6, . We also highlight that the utility of self-attention goes beyond computing internal data representation of a model to improve performance [122] or providing meaningful explanations [52, 141]. In addition, self-attention can also be used to integrate simple and intuitive data-driven techniques into deep learning frameworks.

3.5 Conclusion

We have proposed a task-aware reconstruction technique to improve end-task performance for a time series. In particular, we use attention score distribution to identify timestamps important to an end task. We then sample from those important timestamps and mask them from the data for reconstruction, making the reconstruction *end task-aware*. These tasks are trained alternately, sharing parameters in the same model, thereby enabling the representation learnt through reconstruction to improve end-task performance. Experimental results show that TARNet outperforms the state-of-the-art baselines for both classification and regression tasks. The ablation study highlights the essence of our design choices for the data masking technique, and the case study observations show how TARNet captures the intrinsic task-specific properties of data.

Additional unlabeled data can help to improve TARNet. Although the data reconstruction task is fully unsupervised, it is driven by the end task that requires labeled data. In the future, we wish to explore such task-aware representations under data shift problem and in the presence of outliers.

Chapter 3 incorporates material from the publication “Task-Aware Reconstruction for Time-Series Transformer”, by Ranak Roy Chowdhury, Xiyuan Zhang, Dezhi Hong, Rajesh Gupta, Jingbo Shang, published in Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2022). The dissertation author was the primary investigator and the lead author of this paper.

Table 3.1. Accuracy of ED, MLSTM-FCNs, DTWD, TapNet, and DTWI.

Dataset	ED	MLSTM-FCNs	DTWD	TapNet	DTWI
ArticularyWordRecognition	0.970	0.973	0.987	0.987	0.980
AtrialFibrillation	0.267	0.267	0.220	0.333	0.267
BasicMotions	0.676	0.950	0.975	**1.000**	**1.000**
CharacterTrajectories	0.964	0.985	0.989	**0.997**	0.969
Cricket	0.944	0.917	**1.000**	0.958	0.986
DuckDuckGeese	0.275	0.675	0.600	0.575	0.550
EigenWorms	0.549	0.504	0.618	0.489	-
Epilepsy	0.666	0.761	0.964	0.971	0.978
ERing	0.133	0.133	0.133	0.133	0.133
EthanolConcentration	0.293	0.373	0.323	0.323	0.304
FaceDetection	0.519	0.545	0.529	0.556	-
FingerMovements	0.550	0.580	0.530	0.530	0.520
HandMovementDirection	0.278	0.365	0.231	0.378	0.306
Handwriting	0.200	0.286	0.286	0.357	0.316
Heartbeat	0.619	0.663	0.717	0.751	0.658
InsectWingbeat	0.128	0.167	-	0.208	-
JapaneseVowels	0.924	0.976	0.949	0.965	0.959
Libras	0.833	0.856	0.870	0.850	0.894
LSST	0.456	0.373	0.551	0.568	0.575
MotorImagery	0.510	0.510	0.500	0.590	-
NATOPS	0.850	0.889	0.883	0.939	0.850
PEMS-SF	0.705	0.699	0.711	0.751	0.734
PenDigits	0.973	0.978	0.977	0.980	0.939
Phoneme	0.104	0.110	0.151	0.175	0.151
RacketSports	0.868	0.803	0.803	0.868	0.842
SelfRegulationSCP1	0.771	0.874	0.775	0.652	0.765
SelfRegulationSCP2	0.483	0.472	0.539	0.550	0.533
SpokenArabicDigits	0.967	0.990	0.963	0.983	0.959
StandWalkJump	0.200	0.067	0.200	0.400	0.333
UWaveGestureLibrary	0.881	0.891	0.903	0.894	0.868
PAMAP2	0.718	0.949	0.683	0.865	0.769
OpportunityGestures	0.655	0.768	0.762	0.574	0.715
OpportunityLocomotion	0.845	0.900	0.859	0.850	0.868
Occupancy	0.496	0.873	0.517	0.844	0.526
Total best accuracy	0	0	1	2	1
Average accuracy	0.596	0.651	0.658	0.672	0.675
Ours 1-to-1 Wins	32	26	27	23	31
Ours 1-to-1 Draws	0	0	2	2	1
Ours 1-to-1 Losses	2	8	5	9	2
Mean Rank	12.15	8.79	9.65	7.44	10.44

Table 3.2. Accuracy of NS, WEASEL-MUSE, TS-TCC, TNC, and ShapeNet.

Dataset	NS	WEASEL-MUSE	TS-TCC	TNC	ShapeNet
ArticularyWordRecognition	0.987	0.990	0.953	0.973	0.987
AtrialFibrillation	0.133	0.333	0.267	0.133	0.400
BasicMotions	**1.000**	**1.000**	**1.000**	0.975	**1.000**
CharacterTrajectories	0.994	0.990	0.985	0.967	0.980
Cricket	0.986	**1.000**	0.917	0.958	0.986
DuckDuckGeese	0.675	0.575	0.380	0.460	0.725
EigenWorms	0.878	**0.890**	0.779	0.840	0.878
Epilepsy	0.957	**1.000**	0.957	0.957	0.987
ERing	0.133	0.133	0.904	0.852	0.133
EthanolConcentration	0.236	0.430	0.285	0.297	0.312
FaceDetection	0.528	0.545	0.544	0.536	0.602
FingerMovements	0.540	0.490	0.460	0.470	0.580
HandMovementDirection	0.270	0.365	0.243	0.324	0.338
Handwriting	0.533	**0.605**	0.498	0.249	0.451
Heartbeat	0.737	0.727	0.751	0.746	0.756
InsectWingbeat	0.160	-	0.264	0.469	0.250
JapaneseVowels	0.989	0.973	0.930	0.978	0.984
Libras	0.867	0.878	0.822	0.817	0.856
LSST	0.558	0.590	0.474	0.595	0.590
MotorImagery	0.540	0.500	0.610	0.500	0.610
NATOPS	**0.944**	0.870	0.822	0.911	0.883
PEMS-SF	0.688	-	0.734	0.699	0.751
PenDigits	0.983	0.948	0.974	0.979	0.977
Phoneme	0.246	0.190	0.252	0.207	**0.298**
RacketSports	0.862	0.934	0.816	0.776	0.882
SelfRegulationSCP1	0.846	0.710	0.823	0.799	0.782
SelfRegulationSCP2	0.556	0.460	0.533	0.550	0.578
SpokenArabicDigits	0.956	0.982	0.970	0.934	0.975
StandWalkJump	0.400	0.333	0.333	0.400	0.533
UWaveGestureLibrary	0.884	0.916	0.753	0.759	0.906
PAMAP2	0.885	0.928	0.942	0.938	0.948
OpportunityGestures	0.689	0.553	0.791	0.821	0.730
OpportunityLocomotion	0.859	0.634	0.881	0.874	0.874
Occupancy	0.817	0.556	0.865	0.828	0.852
Total best accuracy	2	5	1	0	2
Average accuracy	0.686	0.688	0.692	0.693	0.717
Ours 1-to-1 Wins	23	25	28	29	25
Ours 1-to-1 Draws	2	3	1	1	2
Ours 1-to-1 Losses	9	6	5	4	7
Mean Rank	7.59	7.79	9.03	9.41	5.47

Table 3.3. Accuracy of TS2Vec, Rocket, MiniRocket, TST, and TARNet.

Dataset	TS2Vec	Rocket	MiniRocket	TST	TARNet
ArticularyWordRecognition	0.987	**0.993**	**0.993**	0.947	0.977
AtrialFibrillation	0.200	0.067	0.133	0.533	**1.000**
BasicMotions	0.975	**1.000**	**1.000**	0.925	**1.000**
CharacterTrajectories	0.995	0.991	0.990	0.971	0.994
Cricket	0.972	**1.000**	0.986	0.847	**1.000**
DuckDuckGeese	0.680	0.500	**0.750**	0.300	**0.750**
EigenWorms	0.847	0.650	0.790	0.720	0.420
Epilepsy	0.964	0.986	**1.000**	0.775	**1.000**
ERing	0.874	**0.989**	0.974	0.930	0.919
EthanolConcentration	0.308	**0.450**	0.430	0.337	0.323
FaceDetection	0.501	0.638	0.612	0.625	**0.641**
FingerMovements	0.480	0.520	0.550	0.590	**0.620**
HandMovementDirection	0.338	0.486	0.392	**0.675**	0.392
Handwriting	0.515	0.596	0.520	0.359	0.281
Heartbeat	0.683	0.741	0.771	**0.782**	0.780
InsectWingbeat	0.466	0.179	0.229	**0.687**	0.137
JapaneseVowels	0.984	0.978	0.986	**0.995**	0.992
Libras	0.867	0.906	0.922	0.861	**1.000**
LSST	0.537	0.635	0.653	0.576	**0.976**
MotorImagery	0.510	0.460	0.610	0.610	**0.630**
NATOPS	0.928	0.872	0.933	0.939	0.911
PEMS-SF	0.682	0.832	0.809	0.930	**0.936**
PenDigits	**0.989**	0.981	0.967	0.981	0.976
Phoneme	0.233	0.273	0.291	0.111	0.165
RacketSports	0.855	0.901	0.868	0.796	**0.987**
SelfRegulationSCP1	0.812	0.867	0.915	**0.961**	0.816
SelfRegulationSCP2	0.578	0.555	0.506	0.604	**0.622**
SpokenArabicDigits	0.988	0.997	0.963	**0.998**	0.985
StandWalkJump	0.467	0.467	0.333	**0.600**	0.533
UWaveGestureLibrary	0.906	**0.931**	0.785	0.913	0.878
PAMAP2	0.941	0.931	0.962	0.948	**0.974**
OpportunityGestures	0.771	0.813	0.809	0.732	**0.830**
OpportunityLocomotion	0.842	0.875	0.886	0.907	**0.908**
Occupancy [49]	0.876	0.832	0.878	0.881	**0.883**
Total best accuracy	1	6	4	7	**17**
Average accuracy	0.722	0.732	0.741	0.745	**0.772**
Ours 1-to-1 Wins	24	20	21	20	-
Ours 1-to-1 Draws	0	2	4	0	-
Ours 1-to-1 Losses	10	12	9	14	-
Mean Rank	7.18	5.18	4.71	5.74	**4.00**

Table 3.4. Root Mean Squared Error (RMSE) of 1-NN-DTWD, 1-NN-ED, 5-NN-ED, and 5-NN-DTWD.

Dataset	1-NN-DTWD	1-NN-ED	5-NN-ED	5-NN-DTWD
AppliancesEnergy	6.036	5.231	4.227	4.019
BenzeneConcentration	4.983	6.535	5.844	4.868
BeijingPM10	139.134	139.229	115.502	115.502
BeijingPM25	88.256	88.193	74.156	72.717
LiveFuelMoisture	57.111	58.238	46.331	46.290
IEEEPPG	37.140	33.208	27.111	33.572
Total best loss	0	0	0	0
Ours 1-to-1 Wins	6	6	6	6
Ours 1-to-1 Draws	0	0	0	0
Ours 1-to-1 Losses	0	0	0	0
Mean Rank	12.167	11.833	8.833	8.833
Avg.Rel.Diff.Mean	0.355	0.379	0.153	0.125

Table 3.5. Root Mean Squared Error (RMSE) of SVR, ResNet, FCN, and Rocket.

Dataset	SVR	ResNet	FCN	Rocket
AppliancesEnergy	3.457	3.065	2.865	2.299
BenzeneConcentration	4.790	4.061	4.988	3.360
BeijingPM10	110.574	95.489	94.438	120.057
BeijingPM25	75.734	64.462	59.726	62.769
LiveFuelMoisture	43.021	51.632	47.877	41.829
IEEEPPG	36.301	33.150	34.325	36.515
Total best loss	0	0	0	0
Ours 1-to-1 Wins	6	6	5	6
Ours 1-to-1 Draws	0	0	0	0
Ours 1-to-1 Losses	0	0	1	0
Mean Rank	8.000	7.333	7.000	6.500
Avg.Rel.Diff.Mean	0.097	0.006	0.022	-0.047

Table 3.6. Root Mean Squared Error (RMSE) of Inception, RF, XGB, TST, and TARNet.

Dataset	Inception	RF	XGB	TST	TARNet
AppliancesEnergy	4.435	3.455	3.489	2.375	**2.173**
BenzeneConcentration	1.584	0.855	0.637	<u>0.494</u>	**0.481**
BeijingPM10	96.749	94.072	93.138	**86.866**	<u>90.482</u>
BeijingPM25	62.227	63.301	<u>59.495</u>	**53.492**	60.271
LiveFuelMoisture	51.539	44.657	44.295	43.138	**41.091**
IEEEPPG	**23.903**	32.109	31.487	27.806	<u>26.372</u>
Total best loss	1	0	0	<u>2</u>	**3**
Ours 1-to-1 Wins	5	6	5	4	-
Ours 1-to-1 Draws	0	0	0	0	-
Ours 1-to-1 Losses	1	0	1	2	-
Mean Rank	6.500	5.500	4.333	<u>2.500</u>	**1.833**
Avg.Rel.Diff.Mean	-0.107	-0.171	-0.196	<u>-0.302</u>	**-0.313**

Table 3.7. Ablation study of TARNet

	TARNet-Random	TARNet-Top μ	TARNet
	Results on 34 classification datasets		
Total best accuracy	6	<u>9</u>	**31**
Average accuracy	<u>0.752</u>	0.741	**0.772**
Ours 1-to-1 Wins	28	25	-
Ours 1-to-1 Draws	5	7	-
Ours 1-to-1 Losses	1	2	-
Mean Rank	2.206	<u>2.176</u>	**1.088**
	Results on 6 regression datasets		
Total best loss	0	<u>1</u>	**5**
Ours 1-to-1 Wins	6	5	-
Ours 1-to-1 Draws	0	0	-
Ours 1-to-1 Losses	0	1	-
Mean Rank	2.667	<u>2.167</u>	**1.167**
Avg.Rel.Diff.Mean	0.046	<u>0.014</u>	**-0.060**

Chapter 4

Sensor Context Aware Self-supervision

In the previous chapter, we focused on task-aware representation learning to enhance end-task performance. Now, we explore Zero-Shot Learning (ZSL), a specific and challenging learning scenario within the fine-tuning paradigm. Unlike traditional fine-tuning, ZSL addresses the limitation of models that struggle to generalize across diverse classes. This chapter introduces ZSL techniques specifically for Human Activity Recognition (HAR) to overcome these challenges and improve model generalization.

HAR identifies human movements from motion sensor data, crucial for mobile and wearable sensing. However, existing HAR systems are trained on a highly restrictive set of motions. Hence, they fail to generalize to diverse human motions, prompting Zero-Shot Learning (ZSL). ZSL for HAR requires auxiliary information about activities, often available only in text form, which presents two key challenges: 1) text and motion sensor data are in different modalities that must be jointly learned, and 2) text-based auxiliary information, like activity names, are static description of an activity that do not provide sufficient fine-grained knowledge about the sequence of bio-mechanical episodes that constitute an activity.

We propose ZeroHAR, a novel two-stage framework for Motion Sensor-based Zero-Shot HAR [1]. In Stage I, we exploit rich contextual sensor name and placement information in complement with sensor readings to contrastively align them with text. In Stage II, we first design prompts to generate precise, fine-grained, textual knowledge about human motions from

[1]Code is publicly available at https://github.com/ranakroychowdhury/ZeroHAR

a Large Language Model (LLM). Finally, we tune the pre-initialized model from Stage I with the target activity descriptions. We compared ZeroHAR with 8 baselines on 5 benchmark HAR datasets, containing a wide range of sensors, placements and activities. ZeroHAR resulted in an 19.1% and 20.4% average improvement in Zero-Shot Accuracy and Macro-F1, respectively, over the next best baseline results, owing to its strongly aligned latent space learnt during Stage I and the fine-grained activity knowledge infused in Stage II.

4.1 Problem Statement

Inertial HAR predicts human activities using data from Inertial Measurement Unit (IMU). Existing inertial HAR systems are trained on data from a limited set of motions, collected and annotated within a controlled laboratory setting. However, these models fail to recognize the richer and more diverse set of motions that humans exhibit in the real world. Annotating vast amounts of IMU data for all possible human movements to train a HAR model is not plausible. Hence, in HAR, although a model may be trained on a limited set of activities, we expect them to recognize unseen activities after deployment, a process known as Zero-Shot Learning (ZSL).

ZSL involves training a model on the data from seen classes and evaluating it on the test data from unseen classes. To achieve this objective, ZSL is trained to learn fine-grained attributes shared among classes that can be generalized to recognize unseen classes. For example, a model may be trained on IMU data for "walking" but is expected to recognize "running" during test time. Both these activities share several similarities in terms of body posture and limb movements. For example, both involve an upright posture with a straight spine and head facing forward, the arms swing alternately with the legs, and they both require a cyclic movement pattern of the legs. Such basic body movements constitute activities at large. Hence, a model that learns these low-level fine-grained knowledge from the activities that it were trained on, can recognize unseen activities at test time.

Existing work in ZSL for Inertial HAR can be categorized into three types, shown in

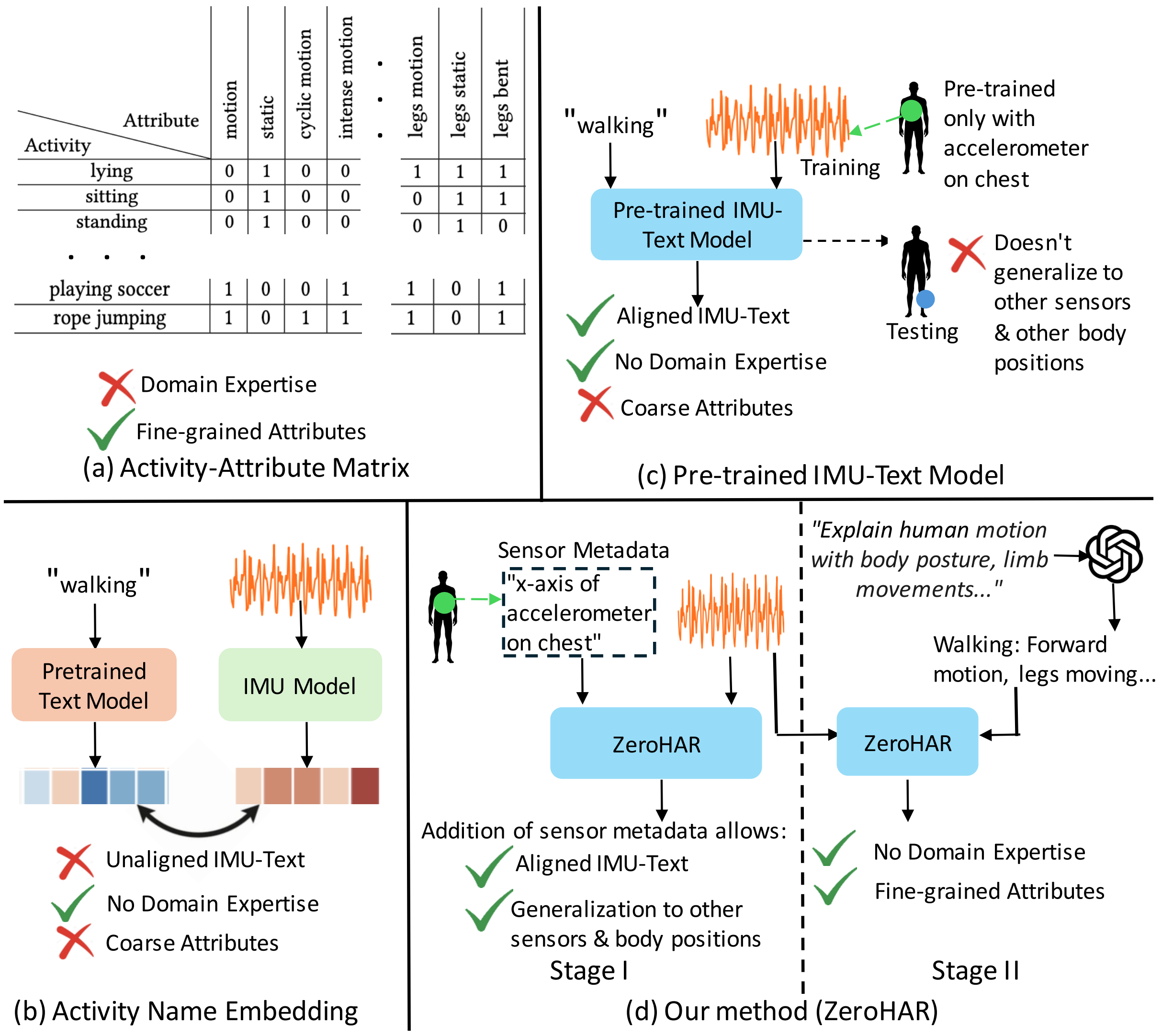
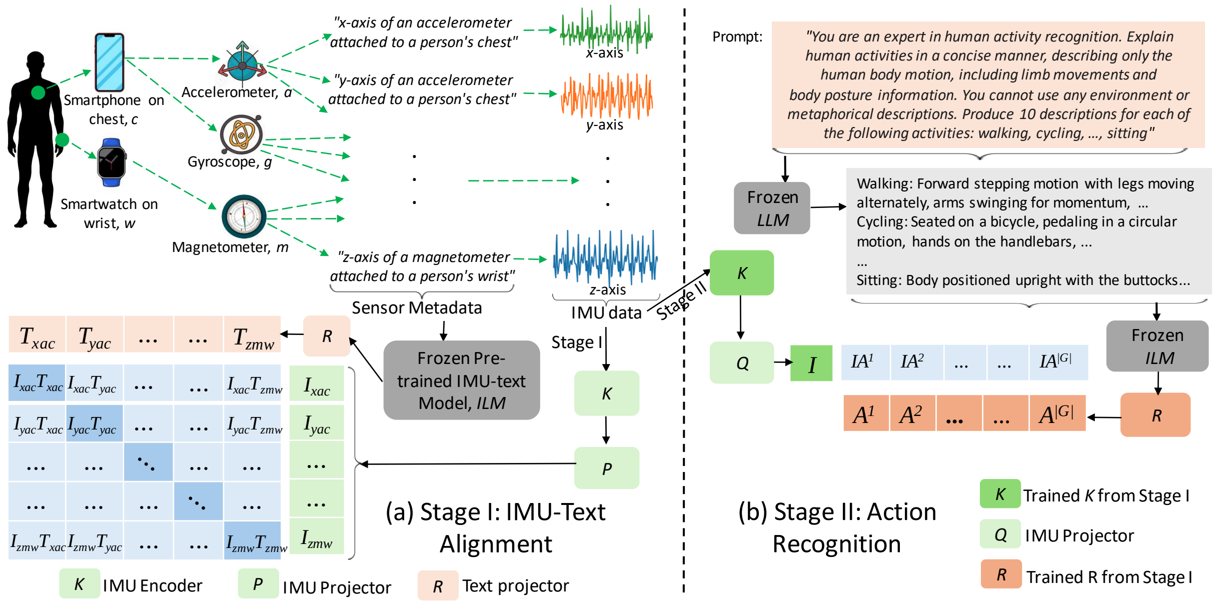


Figure 4.1. Different Approaches to Zero-shot Wearable HAR.

Figure 4.1: **(a)** Manually crafting activity-attribute matrices that encode basic limb movements and body posture information in a binary matrix [126, 19, 18]: This method demands extensive domain knowledge and is time-consuming to design. It is also difficult to come up with a comprehensive set of attributes that can capture the wide variety of human motions, making them impractical for large-scale activity recognition. **(b)** Extracting embeddings of HAR class names from language models and computing their cosine similarity with that of IMU embeddings [81, 133]: This approach mitigates the need for domain expertise but the activity name embeddings lack sufficient fine-grained knowledge necessary for generalization to unseen



Smartphone on chest, c
Smartwatch on wrist, w
Accelerometer, a
Gyroscope, g
Magnetometer, m
"x-axis of an accelerometer attached to a person's chest"
"y-axis of an accelerometer attached to a person's chest"
"z-axis of a magnetometer attached to a person's wrist"
x-axis
y-axis
z-axis
Sensor Metadata
IMU data
Stage I
Stage II
Frozen Pre-trained IMU-text Model, ILM
(a) Stage I: IMU-Text Alignment
K IMU Encoder
P IMU Projector
R Text projector
Prompt:
"You are an expert in human activity recognition. Explain human activities in a concise manner, describing only the human body motion, including limb movements and body posture information. You cannot use any environment or metaphorical descriptions. Produce 10 descriptions for each of the following activities: walking, cycling, ..., sitting"
Frozen LLM
Walking: Forward stepping motion with legs moving alternately, arms swinging for momentum, ...
Cycling: Seated on a bicycle, pedaling in a circular motion, hands on the handlebars, ...
Sitting: Body positioned upright with the buttocks...
Frozen ILM
(b) Stage II: Action Recognition
K Trained K from Stage I
Q IMU Projector
R Trained R from Stage I

Figure 4.2. ZeroHAR Training Overview.

activities. For example, the embedding of "walking" will not include details about how the body posture and limb movements evolve during the activity. Moreover, activity names can have multiple meanings, like "*walk* in the park" introducing ambiguity in the embeddings. Besides, IMU and text are in different modalities and reconciling them poses a significant challenge. **(c)** Developing pre-trained IMU-text models, like ImageBind [42] and IMU2CLIP [86], addresses the alignment problem. However, these models are pre-trained on datasets with specific IMU sensors positioned at specific body locations, limiting their adaptability. Hence, they under-perform when applied to datasets with alternative IMU sensors and varied body placements.

To generalize to all IMU sensors located at any body position and to better align IMU with text for Zero-Shot Inertial HAR, we propose ZeroHAR.

Figure 4.2 shows ZeroHAR's two-stage training setup. In Stage I (IMU-Text Alignment), we compliment each IMU measurement with its corresponding sensor metadata provided as text. The metadata includes information about the measurement axis, the type of sensor measurement, and the body position where this sensor is located. I_{xac} and T_{xac} are embeddings for IMU and text-based sensor knowledge, respectively, from the $x-$axis of accelerometer, a, at chest, c. $I_{xac}T_{xac}$ is the dot product between I_{xac} and T_{xac} for Multimodal Contrastive Learning. ZeroHAR is trained on this IMU and text-based sensor metadata through multimodal contrastive loss that brings the latent space of IMU representation closer to its corresponding sensor information. Encoding such specific knowledge about sensor type and its placement enables ZeroHAR to generalize to different IMU datasets given specific sensor metadata information. Moreover, since the metadata is in text, the joint IMU-text training also helps to align IMU with text. This alignment is important because in the next stage the model needs to learn to map IMU to text-based activity descriptions. In Stage II, we prompt an LLM to generate precise, fine-grained bio-mechanical information about human activity. This provides fine details about human activities, obviating the need to custom design activity-attribute matrices. The pre-initialized model from Stage I is then trained to recognize actions from an IMU with its corresponding activity description as the label in Stage II (Action Recognition). I and A^1 are embedding vectors for a given IMU and an Activity, respectively. IA_1 is the dot product between I and A_1 for Cross-Entropy Loss. This stage aligns various time series measurements of the core body movements (such as arm up or leg kicking) to its text counterparts that constitute activities at large, even for the unseen ones. K, P, Q, and R are learnable blocks.

We extensively evaluate ZeroHAR with 8 baselines on 5 benchmark HAR datasets, encompassing a wide variety in the number and type of IMU sensors and the range of human motions. ZeroHAR resulted in a 19.1% and 20.4% average improvement in Zero-Shot Accuracy and Macro-F1, respectively, over the 2^{nd} best results. We also conduct ablations and case study to show how the warm initialization of the IMU-text latent space during the alignment stage helps ZeroHAR to generalize to unseen classes. Our main contributions are as follows:

- We use sensor metadata to align IMU with text and to generalize our approach to new IMU sensors at different body positions.

- We propose a two-stage training framework for ZSL on Inertial HAR: warm initialization of IMU-text latent space, followed by activity recognition.

- We automate the generation of precise fine-grained activity description from LLM through carefully designed prompts.
- We present new state-of-the-art performance for ZSL on 5 benchmark Inertial HAR datasets.

4.2 Related Works

Zero-Shot Learning (ZSL) extends learned knowledge from a known set of training classes to an unknown set of classes in testing. To do so, it captures shared low-level semantics among classes [109, 45]. One-hot encoding of classes is inadequate to represent such semantics. Some ZSL studies for Inertial HAR manually designs class-attribute matrix, including body posture and limb motion details. [126, 19, 18] (Figure 4.1(a)). While insightful, manual design is impractical for large-scale HAR due to the diverse range of human movements.

Others represented class names as embeddings to learn their auxiliary semantic space [81, 133] (Figure 4.1(b)), mitigating the need for domain expertise. However, this approach faces two challenges: 1) Activity names lack fine-grained motion details, hindering generalization to unseen activities. For instance, a class-attribute matrix may capture “body remains upright” while “standing,” but the embedding of “standing” may lack this specificity. Without learning such fundamental bio-mechanics, a model may struggle to generalize to similar unseen activities like “walking”, where the body also stays upright. Moreover, activity names can have multiple meanings, like “*walk* in the park” or “software is *running* smooth”, leading to ambiguity in embeddings. 2) IMU and textual activity names belong to different modalities, complicating the learning of a shared multimodal space.

To address the limited detail activity names provide about human motion, most Zero-Shot Inertial HAR works utilize a third modality of data, along with IMU and textual activity names, for *both* training and testing. Some use images or videos [119, 26], LiDAR and radar [152],

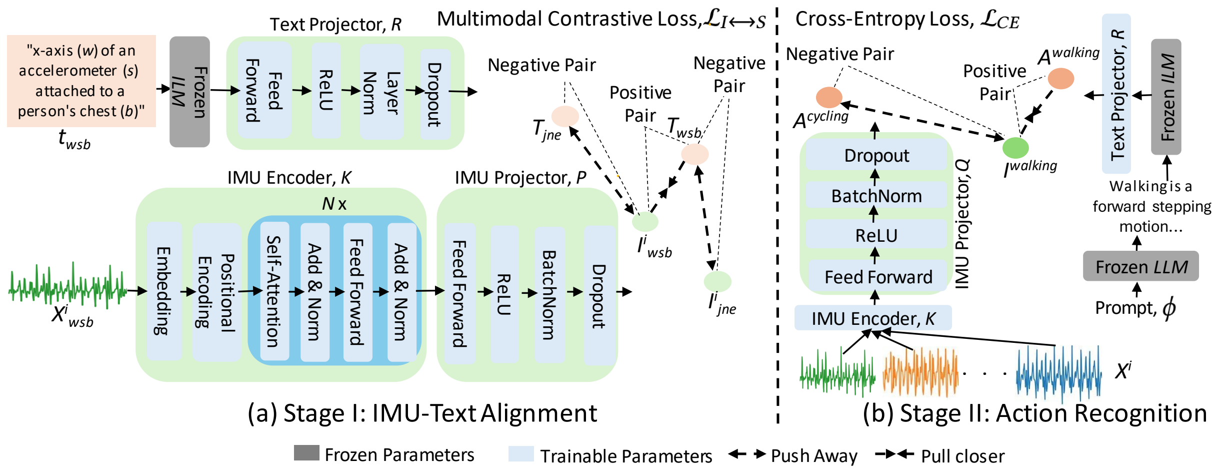


Figure 4.3. ZeroHAR Architectural and Loss Overview.

skeleton [20, 65], or audio [151] as the third modality. However, most IMU datasets lack paired data in another modality, preventing training and testing on IMU-only datasets.

Recently, pre-trained IMU-text models like ImageBind [42] and IMU2CLIP [86] used IMU data and text-based activity names to learn a shared multimodal space. However, these models are pre-trained on datasets with specific IMU sensors located at fixed body positions. They generalize poorly when tested with other IMU sensors at different body positions (Figure 4.1(c)).

ZeroHAR leverages fine-grained information from an LLM to enrich HAR without relying on paired data from a third modality. This includes specific details about body posture and limb motion evolution during activities. This circumvents domain expertise and enables scalable deployment for large-scale HAR, as textual information is more easily accessible than other modalities like videos or skeleton data. Moreover, to enhance IMU-text based HAR, we pre-initialize the IMU-text latent space through contrastive training of IMU data with textual sensor metadata, driving the development of our two-stage framework. Additionally, ZeroHAR seamlessly adapts to new IMU datasets with different sensor types and body positions, provided relevant sensor metadata is available.

4.3 ZeroHAR Framework

In Figure 4.2, we show a schematic diagram of ZeroHAR. In this section, we first present the problem setting, followed by our two-stage training recipe, namely Stage I: IMU-Text Alignment (i.e., Figure 4.2(a)) and Stage II: Action Recognition (i.e, Figure 4.2(b)).

Figure 4.3 dissects the different model components and loss computation of ZeroHAR. (a) Stage I is trained through Multimodal Contrastive Learning between IMU, I_{wsb}, and textual sensor metadata, T_{wsb}, embeddings. (b) Stage II is trained using Cross Entropy Loss between IMU, I, and textual activity description, A, embeddings.

4.3.1 Problem Statement

Let O and U be the set of observed and unobserved activities, respectively. The two sets are disjoint, i.e. $O \cap U = \emptyset$. Let $G = O \cup U$ be the set of all activities, and g denote a given activity, $g \in G$. All the labeled instances for training are from the observed activities in O. More formally, the training set is $D_{tr} = \{(X^i, Y^i) | i = 1, 2, 3, ..., N\}$ where N is the number of training data points, X^i is a multi-variate time series, and $Y^i \in O$ is the activity label corresponding to X^i. In ZSL, the data and labels for the test set come only from the *unobserved* classes: $D_{te} = \{(X^i, Y^i) | i = 1, 2, 3, ..., Z\}$, where Z is the number of test data points and $Y^i \in U$.

Let B be the set of body positions with wearable devices, and b denote a given body position, $b \in B$. Let M_b denote the set of IMU sensors at body position b. Then M denotes the set of IMU sensors at all body positions, hence $M = \{M_1, M_2, ..., M_b, \ldots, M_{|B|}\}$. Hence, the total number of IMU sensors attached to the body is $\sum_{b=1}^{|B|} |M_b|$. If s denotes a given IMU sensor, then M_{sb} denotes IMU sensor s at body position b. Let W denote the set of axes along which an IMU records measurements and w denote a given axis, $w \in W$. All IMUs record data along $x-$, $y-$, and $z-$ axes, so W is constant for every IMU sensor, $W = \{x, y, z\}$, $|W| = 3$. So the total number of channels in a given data point is $|W| \sum_{b=1}^{|B|} |M_b|$. If l is the number of timestamps in the data, then $X^i \in \mathbb{R}^{|W| \sum_{b=1}^{|B|} |M_b| \times l}$. And $X^i_{wsb} \in \mathbb{R}^l$ denote a uni-variate time series from axis, w, of IMU sensor, s, located at body position, b, for data X^i. All notations are summarized in Table **??** under Section C.0.1.

4.3.2 Stage I: IMU-Text Alignment

Sensor Metadata Construction Textual descriptions of human motion biomechanics, instead of activity names, offers finer details crucial for generalization to unseen classes in ZSL. However, integrating text and IMU data, which belong to different modalities, requires reconciling them for effective cross-modal understanding.

To address this challenge, we propose complementing IMU data with corresponding contextual text-based sensor metadata. By training a model with this combined data, we aim to enhance cross-modal alignment, leading to improved downstream classification. Smart devices worn on various body positions, b (e.g., chest or waist pocket for a phone, right or left wrist for a watch) may house multiple IMU sensors, s (e.g., accelerometers, gyroscopes, magnetometers) capturing time-series measurements across multiple axes, w (x, y, and z). Despite the availability of such contextual information in most IMU datasets, it remains underutilized in HAR systems.

Algorithm 4 outlines the IMU-text alignment procedure. For each IMU time series, we construct its text metadata, t_{wsb}, that corresponds to: "$w-$axis of s attached to b" (Line 10). Complimenting an IMU time series with corresponding sensor metadata provides rich contextual information that helps to better align IMU with text.

Alignment Training We embed t_{wsb} using the text encoder of ImageBind [42], a Pre-trained IMU-text Model (*ILM*). Unlike BERT [64], GPT [94], or other language models (*LM*), an *ILM* is pre-trained with both IMU and text. Hence, *ILM*'s text embedding better aligns with IMU than that of *LM*'s. To preserve the IMU-text alignment that *ILM* exhibits, we freeze its parameters. The output of *ILM* is then passed through a text projection block, R, to extract sensor metadata embedding output, T_{wsb}, where $T_{wsb} \in \mathbb{R}^h$ and h is the hidden dimension of the model (Line 11).

ILM's IMU encoder cannot be used on all IMU datasets because their encoder is trained on specific IMU sensors located at specific body positions. It does not generalize well to other IMU sensors located at different body positions. Hence, we use a separate IMU encoder, K, with a transformer backbone [144] (Figure 4.3(a)). The uni-variate IMU data, X^i_{wsb}, is passed through K, followed by an IMU projection block, P, to extract IMU embeddings I^i_{wsb}, $I^i_{wsb} \in \mathbb{R}^h$ (Line 12). Projection blocks R and P are learnable non-linear layers that project text and IMU, respectively, into a shared, latent space.

Cross-Modal Contrastive Learning After obtaining T_{wsb} and I^i_{wsb}, we propose modality-mutual learning for IMU-text alignment. This involves a joint optimization process using a contrastive strategy to refine parameters in both language extraction and sensor encoders. The goal is to align the latent space of IMU embeddings with their corresponding textual sensor metadata embeddings while maintaining separation from unrelated sensor metadata. Contrastive learning [15, 55] pulls similar points closer (anchor and positive) while pushing dissimilar points away (anchor and negative), facilitating good representation learning. We utilize Cross-Modal Contrastive Multiview Coding (CMC) [117] to achieve similar representation learning capability across different modalities, maximizing the similarity between IMU embedding and its corresponding sensor metadata embedding while minimizing the similarity between all other pairs of embeddings via Information Noise Contrastive Estimation (InfoNCE) [89].

For data point X^i, the IMU embedding from w axis of s sensor at body position b, I^i_{wsb}, and its corresponding text embedding, T_{wsb}, are considered a positive pair. To compute the IMU-to-Text Loss, $L_{I^i_{wsb}\to T_{wsb}}$, I^i_{wsb} and T_{wsb} are the anchor and positive, respectively. (I^i_{wsb}, T_{wsb}) forms a positive pair. All other combinations of projections from different sensor channels (I^i_{wsb}, T_{jne}) are treated as negative pairs, where $j = \{1,2,...,|W|\}, n = \{1,2,...,|M_e|\}, e = \{1,2,...,|B|\}$ and $(j \neq w \wedge n \neq s \wedge e \neq b)$. For example, as shown in Figure 4.3(a), if I^i_{wsb} is the anchor, then T_{wsb} is its positive and T_{jne} is one of the $|W|\sum_{b=1}^{|B|}|M_b| - 1$ negatives. $L_{I^i_{wsb}\to T_{wsb}}$ for X^i is calculated as,

$$L_{I^i_{wsb}\to T_{wsb}} = -\log \frac{\exp(sim(I^i_{wsb}, T_{wsb}))/\tau}{\sum_{e=1}^{|B|}\sum_{n=1}^{|M_e|}\sum_{j=1}^{|W|}\exp(sim(I^i_{wsb}, T_{jne}))/\tau}, \tag{4.1}$$

where we use cosine similarity as our similarity function for $sim(I^i_{wsb}, T_{wsb})$. The numerator computes the similarity score between IMU and and its corresponding sensor metadata embedding while the denominator considers similarities with all available sensor metadata. τ is a temperature parameter to scale the similarities.

Similarly, to compute the Text-to-IMU Loss, $L_{T_{wsb} \rightarrow I^i_{wsb}}$, for data point X^i, (I^i_{wsb}, T_{wsb}) forms a positive pair. All other combinations of projections from different input instances (I^i_{jne}, T_{wsb}) are treated as negative pairs, where $j = \{1, 2, ..., |W|\}, n = \{1, 2, ..., |M_e|\}, e = \{1, 2, ..., |B|\}$ and $(j \neq w \land n \neq s \land e \neq b)$. In Figure 4.3(a), if T_{wsb} is the anchor, then I^i_{wsb} is its positive and I^i_{jne} is one of the $|W| \sum_{b=1}^{|B|} |M_b| - 1$ negatives. So $L_{T_{wsb} \rightarrow I^i_{wsb}}$ is calculated as,

$$L_{T_{wsb} \rightarrow I^i_{wsb}} = -\log \frac{\exp(sim(I^i_{wsb}, T_{wsb}))/\tau}{\sum_{e=1}^{|B|} \sum_{n=1}^{|M_e|} \sum_{j=1}^{|W|} \exp(sim(I^i_{jne}, T_{wsb}))/\tau} \tag{4.2}$$

$L_{I^i_{wsb} \rightarrow T_{wsb}}$ and $L_{T_{wsb} \rightarrow I^i_{wsb}}$ are computed in Line 13. Lines 6 - 13 is parallelized by matrix vectorization for efficient computation.

The total IMU-to-Text Contrastive Loss, $L_{I^i \rightarrow T}$, and Text-to-IMU Contrastive Loss, $L_{T \rightarrow I^i}$, over training instance i are computed as,

$$L_{I^i \rightarrow T} = \frac{1}{|W| \sum_{b=1}^{|B|} |M_b|} \sum_{b=1}^{|B|} \sum_{s=1}^{|M_b|} \sum_{w=1}^{|W|} L_{I^i_{wsb} \rightarrow T_{wsb}} \tag{4.3}$$

$$L_{T \rightarrow I^i} = \frac{1}{|W| \sum_{b=1}^{|B|} |M_b|} \sum_{b=1}^{|B|} \sum_{s=1}^{|M_b|} \sum_{w=1}^{|W|} L_{T_{wsb} \rightarrow I^i_{wsb}} \tag{4.4}$$

The total Cross-Modal Contrastive Loss for X^i, $L_{I^i \longleftrightarrow T}$, is computed as the average of the two as follows,

$$L_{I^i \longleftrightarrow T} = \frac{1}{2}(L_{I^i \rightarrow T} + L_{T^i \rightarrow I}) \tag{4.5}$$

$L_{I^i \rightarrow T}$ and $L_{T \rightarrow I^i}$ are computed in Lines 14 and 15, respectively. $L_{I^i \longleftrightarrow T}$ updates model components, K, R, and P, in Line 16. The loss minimization ensures high similarity scores for correct

IMU and textual sensor metadata pairs. This bridges the gap between IMU and text, facilitating cross-modal understanding by joint training.

Example 4 Consider a person with a smartphone on his chest, *c*, equipped with an accelerometer, *a*, and a gyroscope sensor, *g*, and a smartwatch on his wrist, *w*, equipped with a magnetometer sensor, *m*. Each sensor records motion data along $x-$, $y-$, and $z-$ axes. The sensor context information corresponding to $x-$axis of an accelerometer attached to the chest would be t_{xac} = *x-axis of an accelerometer attached to the chest.* Suppose the text embedding corresponding to this sensor context is,
$T_{xac} = [0.4273, 0.0324, 0.1358]$
And the IMU embedding corresponding to the particular channel is,
$I_{xac} = [0.8238, 0.09120, 0.0525]$
Let the text embedding corresponding to all sensor context be as follows:
$T_{xac} = [0.4273, 0.0324, 0.1358]$, $T_{yac} = [0.0444, 0.1482, 0.5043]$, $T_{zac} = [0.5350, 0.2158, 0.5222]$,
$T_{xgc} = [0.9174, 0.0888, 0.5477]$, $T_{ygc} = [0.1913, 0.5139, 0.3179]$, $T_{zgc} = [0.7662, 0.6641, 0.4890]$,
$T_{xmw} = [0.1650, 0.8856, 0.1585]$, $T_{ymw} = [0.3684, 0.3520, 0.7317]$, $T_{zmw} = [0.2900, 0.9304, 0.2549]$
The cosine similarity scores of I_{xac} with all the sensor context embeddings are:
$[0.9699, 0.1748, 0.7549, 0.8902, 0.4201, 0.7674, 0.2961, 0.5051, 0.4030]$
Assuming $\tau = 0.1$, putting these cosine similarities in Eq 4.1 gives,
$L_{I_{xac} \to T_{xac}} = 1.8374$
We can compute the above loss for all the other remaining sensor contexts and repeat this procedure for the IMU data across all channels to compute $L_{I \to T}$, as shown in Eq 4.3.

Similarly, let the IMU embeddings corresponding to all the channels be as follows:
$I_{xac} = [0.9293, 0.1199, 0.1764]$, $I_{yac} = [0.8389, 0.0996, 0.4293]$, $I_{zac} = [0.9360, 0.8584, 0.6904]$
$I_{xgc} = [0.6128, 0.8327, 0.9093]$, $I_{ygc} = [0.6189, 0.1975, 0.3877]$, $I_{zgc} = [0.7419, 0.0707, 0.7184]$
$I_{xmw} = [0.2797, 0.2692, 0.6420]$, $I_{ymw} = [0.9928, 0.9789, 0.1807]$, $I_{zmw} = [0.4012, 0.7538, 0.7084]$
The cosine similarity scores of T_{xac} with all the IMU embeddings are:
$[0.9914, 0.9859, 0.8026, 0.6662, 0.9512, 0.8959, 0.6387, 0.7602, 0.5856]$ Putting these cosine similarities in Eq 4.2 gives,
$L_{T_{xac} \to I_{xac}} = 2.0251$
Similarly, we can compute the above loss for the IMU data across all channels and repeat this procedure for the remaining sensor contexts to compute $L_{T \to I}$, as shown in Eq 4.4.

Algorithm 4. Stage I: IMU-Text Alignment

```
Input: $D_{tr}, B, M, W, ILM$
Hyper-parameters: $\tau$
Output: Trained ($K$ and $R$)

1:  $K$, $P$, and $R$ initialized randomly
2:  for $X^i$ in $D_{tr}$ do
3:      for $b$ in $B$ do
4:          for $s$ in $M_b$ do
5:              for $w$ in $W$ do
6:                  $t_{wsb} \leftarrow$ “$w-$axis of $s$ attached to $b$”
7:                  $T_{wsb} \leftarrow R(ILM(t_{wsb}))$, $T_{wsb} \in \mathbb{R}^h$
8:                  $I^i_{wsb} \leftarrow P(K(X^i_{wsb}))$, $I^i_{wsb} \in \mathbb{R}^h$
9:                  Compute $L_{I^i_{wsb} \rightarrow T_{wsb}}$ & $L_{T_{wsb} \rightarrow I^i_{wsb}}$ (Eq. (4.1) & (4.2), respectively)
10:             end for
11:         end for
12:     end for
13:     Compute $L_{I^i \rightarrow T}$ & $L_{T \rightarrow I^i}$ (Eq. (4.3) & (4.4), respectively)
14:     Compute $L_{I^i \longleftrightarrow T}$ from $L_{I^i \rightarrow T}$ and $L_{T \rightarrow I^i}$ (Eq. (4.5))
15:     Update $K$, $P$, and $R$ based on $L_{I^i \longleftrightarrow T}$
16: end for
17: return Trained ($K$ and $R$)
```

4.3.3 Stage II: Action Recognition

With K and R pre-initialized via joint IMU and sensor metadata training, the model has acquired a robust, aligned latent space shared between IMU and text modalities. This will help ZeroHAR to learn to recognize activities from IMU. However, relying solely on activity name embeddings lacks sufficient detail about human movements, potentially hindering generalization to unseen classes and resulting in subpar zero-shot HAR performance.

Activity Description Generation Algorithm 5 outlines our action recognition pipeline. We prompt GPT-4 [90], a Large Language Model (*LLM*), to generate fine-grained description for each activity in G which consists of all observed and unobserved activities. Prompt ϕ is detailed in Figure 4.2(b). Since an activity can be explained in numerous ways, we generate c descriptions per activity to obtain a more stable and less noisy estimate of activity representation. Given the potential for environmental details in LLM-generated descriptions, such as ”walk in the park” or ”sleeping soundly,” we design ϕ to focus strictly on bio-mechanics, body posture, and limb motion, explicitly instructing it to avoid environmental or metaphorical language. The prompt is outlined in Figure 4.2(b). If $\beta_g = \{\beta_{g_1}, \beta_{g_2}, ..., \beta_{g_c}\}$ denotes a set of c generated descriptions for activity g, then $\beta = \{\beta_1, \beta_2, ..., \beta_g, ..., \beta_{|G|}\}$ (Line 7). We manually check all the generated descriptions to verify their accuracy.

Action Recognition Training β is fed through the text encoder of the frozen *ILM* to extract IMU-aligned text embeddings, followed by the pre-initialized text projector, R, from Stage I. The extracted embeddings, α, represent c embeddings corresponding to c descriptions for each activity in G, $\alpha \in \mathbb{R}^{c|G| \times h}$, where h is the hidden dimension of the model (Line 9). We average c embeddings per activity from α to extract A, a single embedding for each activity, $A \in \mathbb{R}^{|G| \times h}$ (Line 10). Representing an activity by the average of c embeddings instead of one, helps in reducing the variance. Similarly, the IMU measurements for data point X^i are also passed through the pre-initialized IMU encoder, K, from Stage I, followed by an IMU projector, Q, which consists of some learnable non-linear layers (Figure 4.3(b)), to extract I^i, $I^i \in \mathbb{R}^h$ (Line 12).

If a is the true activity corresponding to X^i, then (I^i, A^a) represents the corresponding IMU and activity embedding. We compute cross entropy loss, L^i_{CE}, by comparing the cosine similarity of (I^i, A^g), with all the activities in G (Line 13). K, R, and Q, are then trained by backpropagating L^i_{CE} in Line 14.

$$L^i_{CE} = -\log \frac{exp(sim(I^i, A^a))}{\sum_{g=1}^{|G|} exp(sim(I^i, A^g))} \tag{4.6}$$

We use different optimizers to train the IMU-related layers (K, P from Stage I, Q from Stage II) and text-related layers (R in Stage I and II) of ZeroHAR.

Example 5 Consider the IMU embedding for a given instance of walking is,
$I^{walking} = [0.9959, 0.7521, 0.0194]$
Let the global set of activities be $G = walking, sitting, standing, cycling, jumping$ but the training set contain data for only the following activities $O = walking, sitting, standing$.
We use prompt, ϕ = *"You are an expert in human activity recognition. You can explain human activities in various ways. Explain the human activities in a concise manner, describing only the human body motion, including limb movements and body posture information. You cannot use any environment or metaphorical descriptions. Capture the entire activity in a single description. Produce 10 different descriptions for each of the following activities: walking, cycling, standing, sitting, jumping"* to generate 10 descriptions for each activity.
A sample description for *walking* could be *Progressive forward motion with alternating steps, arms swinging in rhythm, and head upright.*
We extract embeddings for all 10 descriptions corresponding to an activity and average them. Suppose, the averaged embedding are as follows:
$A^{walking} = [0.7059, 0.2072, 0.1001]$, $A^{sitting} = [0.6968, 0.9024, 0.5484]$, $A^{standing} = [0.2623, 0.4603, 0.8496]$
The cosine similarity scores of $I^{walking}$ with all the activity embeddings are:
$[0.9289, 0.8760, 0.4992]$
Putting these cosine similarities in Eq 4.6 gives,
$L_{CE} = 0.9552$

Zero-Shot Action Recognition During testing, we predict the activity associated with the largest cosine similarity score,

$$\hat{Y}^i = \underset{g \in U}{\arg\max}\, F(X^i, A^g) \tag{4.7}$$

, where g is a given activity from the set of unobserved activities U, $X^i \in D_{te}$, A^g is the embedding corresponding to activity g, F is ZeroHAR, and $\hat{Y}^i$ is the prediction corresponding to X^i.

Example 6 During testing, we may get an instance of a person cycling whose corresponding IMU embedding is,
$I^{cycling} = [0.9809, 0.2470, 0.3002]$
Along with the averaged embeddings for *walking, sitting,* and *standing* from the last example, we also have the averaged embeddings for *cycling* and *jumping* from the LLM generated descriptions,
$A^{cycling} = [0.6035, 0.6676, 0.9218]$, $A^{jumping} = [0.7384, 0.2264, 0.4257]$

Algorithm 5. Stage II: Action Recognition

Input: D_{tr}, G, LLM, ILM, Trained (K and R) from Stage I
Hyper-parameters: c
Output: Trained (K, Q, and R)

```
1:  $Q$ initialized randomly
2:  $\phi$ constructed from $G$
3:  $\beta \leftarrow LLM(\phi)$
4:  for $(X^i, Y^i)$ in $D_{tr}$ do
5:      $\alpha \leftarrow R(ILM(\beta))$, $\alpha \in \mathbb{R}^{c|G| \times h}$
6:      $A \leftarrow$ Mean $c$ embeddings per class from $\alpha$, $A \in \mathbb{R}^{|G| \times h}$
7:      $A^a \leftarrow$ Embedding for activity, $a$, where $a = Y^i$, $A^a \in \mathbb{R}^h$
8:      $I^i \leftarrow Q(K(X^i))$, $I^i \in \mathbb{R}^h$
9:      Compute $L^i_{CE}$ (Eq. (4.6))
10:     Update $K$, $Q$, and $R$, based on $L^i_{CE}$
11: end for
12: return Trained ($K$, $Q$, and $R$)
```

The cosine similarity scores of $I^{cycling}$ with $A^{walking}, A^{sitting}, A^{standing}, A^{cycling}, A^{jumping}$ are as follows:
$[0.9876, 0.8023, 0.5925, 0.7604, 0.9758]$
Since the highest cosine similarity score corresponds to the text embedding for *walking*, ZeroHAR will mistakenly classify the IMU data as *walking*, even though it actually represents *cycling*.

4.4 Experimental Results

4.4.1 Dataset, Baselines and Metrics

We present the datasets, baselines, metrics, followed by the experimental setup. We show and compare ZeroHAR's Zero-Shot Classification (ZSC) performance with baselines. We perform ablations and a case study to justify our motivation and training approach.

We use 5 Inertial HAR benchmark datasets for evaluation, summarized in Table 4.1. Each dataset has one device at each location specified under "Body Positions" column. Each device is equipped with all the sensors under the "Sensors" column. Each sensor generate time-series data along $x-$, $y-$, and $z-$ axes. The datasets represent a wide variety in the number and type of

Table 4.1. Datasets used. Each body position is equipped with all the sensors specified under sensors column.

Dataset	Body Positions	IMU sensors
Opportunity [98]	back, right upper arm, right lower arm, left upper arm, left lower arm	accelerometer, gyroscope, magnetometer
PAMAP2 [97]	wrist, ankle, chest	accelerometer, gyroscope, magnetometer
Harth [77]	lower back, thigh	accelerometer
USCHAD [145]	front right hip	accelerometer, gyroscope
WISDM [130]	wrist	accelerometer, gyroscope

devices (some have 1 IMU sensor, while others have 3) and their associated body positions. The datasets are as follows:

- **Opportunity**[2] [98] collects readings from 4 users with 6 runs per user. The full dataset includes annotations on multiple levels, and we use mid-level gesture annotations. Sensors include body-worn, object, and ambient sensors. Corresponding activities include opening door, closing fridge, drinking from cup, etc.
- **PAMAP2**[3] [97] collects data from 9 subjects with IMUs sampled at 100 Hz and a heart rate monitor sampled at 9Hz. Three IMUs are positioned over the wrist on the dominant arm, on the chest, and on the dominant side's ankle, respectively.
- **Harth**[4] [77] involves 22 subjects using 2 3-axial accelerometers.
- **USCHAD**[5] [145] involves 14 subjects using MotionNode (6-DOF IMU designed for human motion sensing applications). To complete the study, each subject was asked to engage in each activity 5 times at different locations, both indoors and outdoors, over the course of multiple days.

[2]https://archive.ics.uci.edu/ml/datasets/opportunity+activity+recognition
[3]http://archive.ics.uci.edu/ml/datasets/pamap2+physical+activity+monitoring
[4]https://github.com/ntnu-ai-lab/harth-ml-experiments
[5]https://sipi.usc.edu/had/

Table 4.2. Zero-shot classification on Opportunity, PAMAP2, and Harth.

Baselines	Opportunity [98]		PAMAP2 [97]		Harth [77]	
	Acc	F1	Acc	F1	Acc	F1
	Activity-Attribute based methods					
NuActiv [19]	$39.2_{(8.4)}$	$26.8_{(12.6)}$	$42.7_{(9.3)}$	$36.4_{(11.5)}$	$18.1_{(7.5)}$	$14.6_{(2.1)}$
SemAtt [18]	$53.0_{(9.2)}$	$42.9_{(8.1)}$	$47.2_{(12.1)}$	$37.9_{(10.6)}$	$21.5_{(6.4)}$	$28.1_{(3.1)}$
NCBM [126]	$45.5_{(7.8)}$	$40.6_{(13.3)}$	$\underline{58.8}_{(9.4)}$	$\underline{45.2}_{(12.9)}$	$11.6_{(9.1)}$	$13.7_{(4.8)}$
LETS-GZSL [10]	$31.7_{(12.4)}$	$21.9_{(8.6)}$	$25.9_{(14.5)}$	$18.8_{(11.2)}$	$28.0_{(32.3)}$	$14.0_{(15.5)}$
	Activity Name Embedding based methods					
SemHAR [81]	$56.3_{(11.0)}$	$44.9_{(14.2)}$	$46.7_{(6.7)}$	$31.7_{(8.4)}$	$13.8_{(17.1)}$	$5.7_{(3.6)}$
SHARE [146]	$\underline{61.4}_{(8.0)}$	$\underline{50.8}_{(12.7)}$	$53.3_{(8.7)}$	$35.8_{(10.0)}$	$18.4_{(12.5)}$	$18.1_{(7.8)}$
NonVisual [2]	$52.1_{(15.6)}$	$36.4_{(10.9)}$	$29.9_{(4.4)}$	$15.7_{(3.1)}$	$47.9_{(36.5)}$	$17.2_{(12.9)}$
	Pre-trained IMU-text Model based methods					
ImageB-pre [42]	$34.3_{(4.8)}$	$23.5_{(11.0)}$	$31.9_{(2.7)}$	$29.4_{(4.9)}$	$11.3_{(15.5)}$	$14.1_{(19.2)}$
ImageB-fine [42]	$28.8_{(8.7)}$	$27.6_{(9.1)}$	$37.1_{(7.7)}$	$25.2_{(2.9)}$	$\underline{58.4}_{(31.5)}$	$\underline{31.0}_{(14.5)}$
ZeroHAR	$\mathbf{72.6}_{(6.0)}$	$\mathbf{59.0}_{(11.3)}$	$\mathbf{70.0}_{(2.1)}$	$\mathbf{59.4}_{(3.1)}$	$\mathbf{69.0}_{(12.6)}$	$\mathbf{33.1}_{(6.1)}$

- **WISDM**[6] [130] is collected from IMUs sampled at 20Hz. 51 subjects perform 18 activities for 3 minutes respectively.

We compare ZeroHAR with 3 types of baselines shown in Figure 4.1:

- **NuActiv** [19] develops an developed an active learning algorithm based on class-attribute matrix.
- **SemAtt** [18] proposes a semantic attribute layer that takes into account both the hierarchical and sequential nature of activity data.
- **NCBM** [126] proposes nonlinear compatibility function between feature space instance and custom-designed semantic space prototypes based on class-attribute matrix.
- **LETS-GZSL** [10] combines time-series embedding module with statistical features to project the concatenated features into a latent space to perform Generalized ZSL. We adapt it for ZSL in this work.

[6]https://archive.ics.uci.edu/ml/datasets/WISDM+Smartphone+and+Smartwatch+Activity+and+Biometrics+Dataset+

Table 4.3. Zero-shot classification on USCHAD and WISDM.

Baselines	USCHAD [145]		WISDM [130]	
	Acc	F1	Acc	F1
	Activity-Attribute based methods			
NuActiv [19]	$13.9_{(5.3)}$	$16.9_{(3.5)}$	$18.5_{(7.1)}$	$16.6_{(5.7)}$
SemAtt [18]	$17.2_{(4.9)}$	$20.6_{(9.5)}$	$15.7_{(10.5)}$	$11.6_{(3.6)}$
NCBM [126]	$21.9_{(8.2)}$	$27.4_{(11.4)}$	$21.3_{(2.7)}$	$21.5_{(9.3)}$
LETS-GZSL [10]	$22.4_{(6.4)}$	$13.2_{(5.3)}$	$15.8_{(4.1)}$	$11.1_{(2.5)}$
	Activity Name Embedding based methods			
SemHAR [81]	$26.4_{(10.2)}$	$19.9_{(8.7)}$	$18.5_{(11.7)}$	$15.8_{(9.3)}$
SHARE [146]	$31.1_{(6.7)}$	$24.5_{(6.4)}$	$22.6_{(8.2)}$	$17.9_{(10.5)}$
NonVisualSensors [2]	$29.2_{(12.9)}$	$17.0_{(5.8)}$	$16.2_{(0.5)}$	$4.65_{(0.1)}$
	Pre-trained IMU-text Model based methods			
ImageBind-pre [42]	$25.0_{(7.5)}$	$16.5_{(3.7)}$	$\underline{27.5}_{(3.2)}$	$\underline{21.9}_{(2.8)}$
ImageBind-fine [42]	$\underline{45.3}_{(3.1)}$	$\underline{37.3}_{(4.7)}$	$\mathbf{29.2}_{(1.0)}$	$20.8_{(1.5)}$
ZeroHAR	$\mathbf{61.8}_{(9.6)}$	$\mathbf{50.2}_{(12.1)}$	$26.0_{(10.0)}$	$\mathbf{22.4}_{(9.5)}$

Table 4.4. Regular HAR Classification Performance.

Baselines	Opportunity [98]		Harth [77]		USCHAD [145]	
	Acc	F1	Acc	F1	Acc	F1
TST [144]	78.4	66.8	<u>97.4</u>	<u>50.1</u>	**64.1**	<u>59.4</u>
TARNet [24]	<u>78.9</u>	66.9	96.2	48.1	56.4	53.3
Mini-Rocket [28]	<u>78.9</u>	<u>67.0</u>	89.7	47.2	<u>58.0</u>	**60.1**
ZeroHAR	**79.6**	**75.5**	**98.2**	**50.8**	57.3	54.9

- **SemHAR** [81] proposes word embedding vectors to represent class names, that is used as auxiliary information to do Zero-Shot HAR.
- **SHARE** [146] accounts for shared structures of activity label names by using an encoder for extracting features from input sensory time series and a decoder for generating label names.
- **ImageB-pre** [42] pre-trained model across six different modalities, including IMU and text. We directly use their IMU encoder.
- **ImageB-fine** [42] We fine-tune the IMU encoder by adding a linear classifier on top of the

Table 4.5. Ablation Study of ZeroHAR on Opportunity, PAMAP2, and Harth.

Ablations	Opportunity [98]		PAMAP2 [97]		Harth [77]	
	Acc	F1	Acc	F1	Acc	F1
	ZeroHAR trained only on Stage II, without Stage I					
(1) - w label name	$53.0_{(5.8)}$	$43.6_{(9.5)}$	$48.5_{(8.3)}$	$36.9_{(11.7)}$	$17.5_{(8.1)}$	$14.7_{(6.5)}$
(2) - w act descrip	$55.6_{(13.6)}$	$45.1_{(8.0)}$	$53.1_{(7.3)}$	$41.0_{(4.8)}$	$25.9_{(5.2)}$	$16.2_{(4.1)}$
(3) - (2) & Metadata	$61.3_{(11.7)}$	$46.8_{(7.1)}$	$59.8_{(10.4)}$	$54.4_{(8.7)}$	$23.6_{(9.1)}$	$16.0_{(2.4)}$
	ZeroHAR trained on both Stage I and II					
(4) - w label name	$68.4_{(7.2)}$	$54.2_{(11.0)}$	$\underline{65.5}_{(5.9)}$	$\underline{57.0}_{(8.2)}$	$49.1_{(9.7)}$	$21.8_{(7.8)}$
(5) - w BERT	$\underline{71.1}_{(11.8)}$	$\underline{57.5}_{(6.8)}$	$64.3_{(7.4)}$	$53.7_{(9.3)}$	$\underline{56.2}_{(5.8)}$	$\underline{22.7}_{(4.4)}$
ZeroHAR	$\mathbf{72.6}_{(6.0)}$	$\mathbf{59.0}_{(11.3)}$	$\mathbf{70.0}_{(2.1)}$	$\mathbf{59.4}_{(3.1)}$	$\mathbf{69.0}_{(12.6)}$	$\mathbf{33.1}_{(6.1)}$

Table 4.6. Ablation Study of ZeroHAR on USCHAD and WISDM.

Ablations	USCHAD [145]		WISDM [130]	
	Acc	F1	Acc	F1
	ZeroHAR trained only on Stage II, without Stage I			
(1) - w label name	$23.6_{(9.4)}$	$18.2_{(7.7)}$	$20.7_{(12.6)}$	$15.4_{(5.0)}$
(2) - w act descrip	$30.4_{(11.6)}$	$24.9_{(6.3)}$	$25.0_{(7.6)}$	$18.9_{(8.9)}$
(3) - (2) & Metadata	$27.4_{(8.5)}$	$21.6_{(5.8)}$	$\underline{25.4}_{(7.3)}$	$16.3_{(12.8)}$
	ZeroHAR trained on both Stage I and II			
(4) - w label name	$48.4_{(13.8)}$	$43.9_{(9.3)}$	$21.5_{(3.8)}$	$\underline{21.6}_{(4.6)}$
(5) - w BERT	$\underline{58.3}_{(1.4)}$	$\underline{46.1}_{(6.5)}$	$24.2_{(8.1)}$	$21.3_{(11.7)}$
ZeroHAR	$\mathbf{61.8}_{(9.6)}$	$\mathbf{50.2}_{(12.1)}$	$\mathbf{26.0}_{(10.0)}$	$\mathbf{22.4}_{(9.5)}$

pre-trained model. We then train it on the seen and evaluate on the unseen classes for our datasets.

The activity-attribute based methods define matrices for Opportunity and PAMAP2 datasets, but not for others. We then adapt these attributes to design matrices for the remaining datasets.

Other Zero-Shot HAR works require a third modality paired with IMU and text for *both* train and test, like videos [26], skeleton [20], audio [151]. However, our datasets lack additional paired modalities, thus replicating those baselines was not feasible.

We evaluate the ZSC performance of ZeroHAR and baselines using average per-class

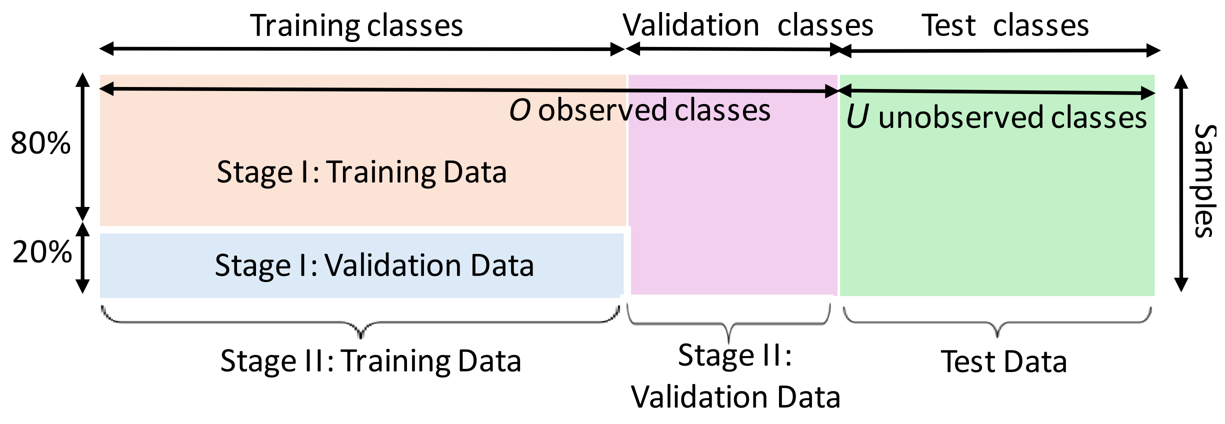


Figure 4.4. Training, validation, and test data splits.

accuracy and macro-F1 score. Macro-F1 is defined as macro-F1 $= \frac{1}{|U|} \sum_{g=1}^{|U|} 2 \times \frac{Prec_g \times Rec_g}{Prec_g + Rec_g}$, where $Prec_g$ and Rec_g, represent the precision and recall for activity g, respectively, and $|U|$ is the total number of unseen activities in the test set, D_{te}.

4.4.2 Experimental Setup

Figure 4.4 shows the train and validation set for Stage I and II of ZeroHAR and the test set. Each row represents #samples and column represent #classes. Note that #samples in each set may be different. The test set comprises novel classes, U, unseen during training. To facilitate early stopping, hyper-parameter selection, and gauge ZeroHAR's performance during testing, evaluation on a validation set is necessary during training. Therefore, we reserve data from certain novel classes within O to form a validation set, O_{va}, at the start of training. Hence, during Stage II training, ZeroHAR only observes data from O_{tr}, where $O_{tr} = O - O_{va}$. We also reserve some data from O_{tr} to use as Stage I validation.

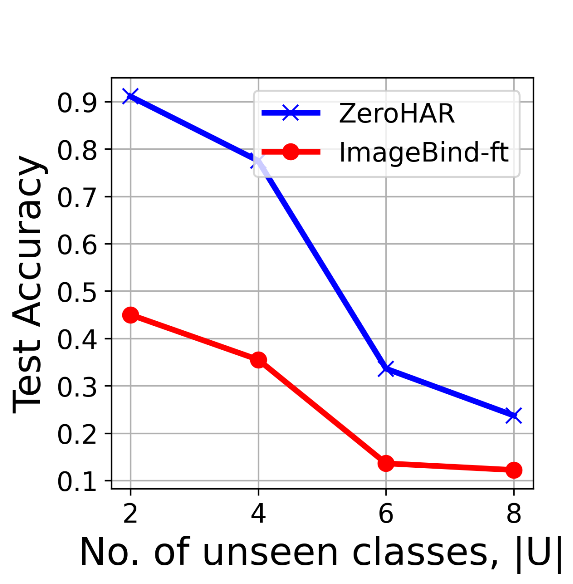
We randomly partition the G classes into O and U for each dataset. To ensure result reliability and assess confidence, we perform this split three times per dataset, (Table C.3 under Section C.0.2). All available classes appear once across the three splits to reflect ZSC performance comprehensively. We conduct three experiments on each dataset, using a different split each time. Specifically, for Experiments 1, 2, and 3, the test classes (U) correspond to fold 1, fold 2, and fold 3, respectively, while their respective training classes (O) are: fold 2 $\cup$ fold 3, fold 1 $\cup$ fold 3, and fold 1 $\cup$ fold 2.

We normalize the datasets for each of our experiments. We train all our baseline models with sufficient hyper-parameter tuning to produce results. Since our datasets are widely heterogeneous in terms of number of data points, sensors, body positions, and sampling frequency, we obtain better performance via cursory tuning of dataset-specific hyper-parameters. We set the temperature parameter τ, in Algorithm 4, to 0.05 and the number of descriptions per activity, c, in Algorithm 5 to 10. We use two Adam optimizers, one to update the IMU modality (IMU Encoder and IMU projectors P and Q for Stage I and II, respectively), and one to update the text modality (text projector R). We use a batch size of 128, learning rate of 0.001, 8 self-attention layers with 8 heads for the IMU Encoder, a dropout of 0.01 and a hidden dimension, h, of 128, for both Stage I and II. We save the model with the lowest validation loss and evaluate it on the test set.

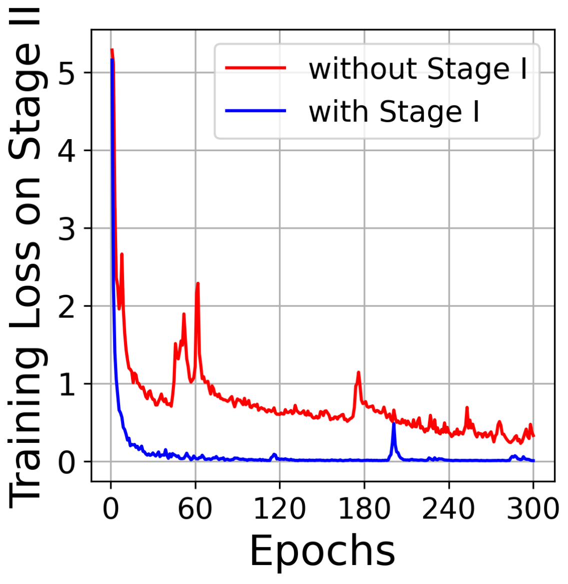
4.4.3 Results

Table 4.2 and 4.3 shows the results. We report the $Mean_{standard\_deviation}$ of Average Per-Class Accuracy and Macro-F1 scores across 3 seen-unseen class splits. Higher Accuracy and Macro-F1 are better. We **bold** the best score and underline the second best.

Overall, Activity-Attribute methods perform poorly, followed by Activity Name Embedding and Pre-trained IMU-text Model. Due to the unavailability of Activity-Attribute matrices for Harth, USCHAD, and WISDM, we used attributes from PAMAP2 and Opportunity [126] to define these matrices. However, the results for these datasets were subpar, indicating that these attributes are too restrictive for generalization to new activities. While Activity Name Embeddings exhibit better discriminative ability, they lack fine-grained information about each activity. Pre-trained models like ImageBind, trained on the Ego4D dataset with sensors attached to the head, perform poorly when directly evaluated for ZSC on our datasets, which feature IMU sensors at various body positions. However, ImageBind fine-tuned on our datasets performs relatively better and serves as the best overall baseline. We compute ZeroHAR's improve-



(a) Test Accuracy vs #unseen classes, $|U|$



(b) Stage II's Loss Convergence

Figure 4.5. Scaling Laws and Loss Convergence of ZeroHAR.

ment over the 2^{nd} best results (underlined values in Table 4.2 and 4.3) shown in Table C.2 (Section C.0.2). ZeroHAR achieves +19.1% and +20.4% relative improvement in Average Accuracy and Macro-F1, respectively, over the 2^{nd} best results.

Performance on General HAR General HAR assumes all activities are observed during training and is a relatively simpler task than ZSC, solvable by simple models. To demonstrate that incorporating relevant contextual information about HAR systems can still yield competitive performance, we compare ZeroHAR with state-of-the-art baselines, like TST [144], TARNet [24], and Mini-Rocket [28]. Table 4.4 shows the accuracy and macro-F1 on Regular HAR Classification. Higher Accuracy and Macro-F1 are better. We **bold** the best score and underline the second best. The results indicate that ZeroHAR outperforms other methods on Opportunity and Harth and is competitive on USCHAD. This underscores the potential of natural language-inspired sensory applications to remain competitive in their core tasks.

Test Accuracy vs no. of unseen classes, $|U|$ To compare how test accuracy on unseen classes vary with the number of unseen classes, $|U|$, we compare ZeroHAR with ImageBind-fine (best baseline as per Table 4.2, 4.3) on fold 1 of Opportunity dataset. Result in Figure 4.5(a) shows that accuracy goes down with increase in $|U|$ for both models, but ZeroHAR surpasses ImageBind-fine for all $|U|$.

Loss Convergence To assess the effect of Stage I (IMU-Text Alignment) on ZeroHAR, we compare its loss convergence when trained on both Stage I and II versus Stage II alone. Figure 4.5(b) shows the results on fold 1 of the Opportunity dataset. ZeroHAR trained with Stage I converges faster and achieves a lower loss during Stage II compared to the model trained solely on Stage II. Stage I involved contrastive training of IMU with textual sensor metadata, providing a warm initialization of the shared space, that simplified subsequent learning in Stage II.

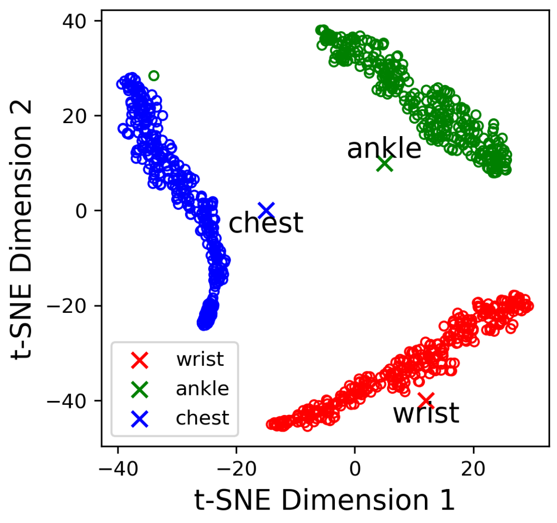
4.4.4 Ablations

We analyze ZeroHAR's performance through various ablations to justify our training decisions. Table 4.5 and 4.6 presents the results. $Mean_{standard\_deviation}$ of Average Per-Class Accuracy and Macro-F1 scores across 3 seen-unseen class splits. Higher Accuracy and Macro-F1 are better. We **bold** the best score and <u>underline</u> the second best.

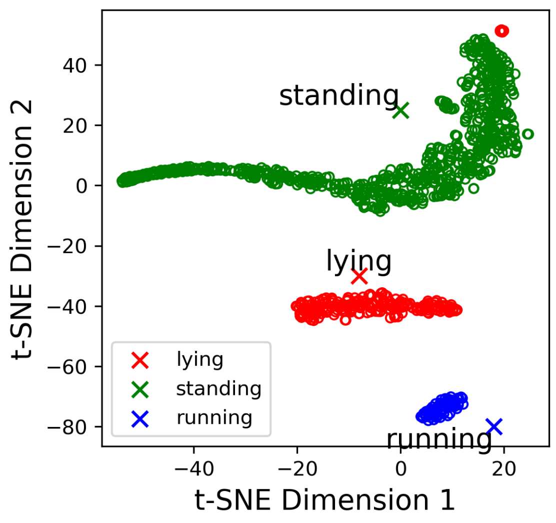
Leveraging activity descriptions (2) improves performance compared to using label names alone (1). Adding sensor metadata to activity descriptions (3) further enhances performance. Training on Stage I: IMU-Text Alignment before Stage II: Action Recognition notably improves performance, validating our hypothesis. Stage I provides a warm initialization to the IMU-text shared latent space, thereby improving the mapping of IMU to their corresponding textual activity descriptions in the subsequent stage. Using activity descriptions (ZeroHAR) yields better results than using label names alone (4). Employing ImageBind for text embedding (ZeroHAR) instead of BERT (5) also improves results due to its alignment with IMU data.

4.4.5 Case Study on IMU-Text Alignment

Figure 4.6 depicts the IMU-text latent space learned by ZeroHAR on PAMAP2, showing 'o' - IMU embeddings with (a) 'x' - embedding of body positions for Stage I and (b) 'x' -



(a) IMU with Body Position - Stage I



(b) IMU with Class Description - Stage II

Figure 4.6. t-SNE vizualization of ZeroHAR on PAMAP2.

embedding of *unseen* test classes' description in fold 3 for Stage II. Figure 4.6(a) demonstrates ZeroHAR's ability to align IMU data from different body parts with their corresponding word embeddings, highlighting that joint training of IMU with its corresponding sensor metadata can bring the latent space of IMU closer to text. Figure 4.6(b) shows that despite not being trained on any data from these unseen classes, ZeroHAR can align their IMU data with their respective textual activity description embeddings.

4.5 Conclusion

We present ZeroHAR, a two stage framework to address the Zero-Shot Learning problem in Inertial HAR. In Stage I: IMU-Text Alignment, we compliment IMU with text-based contextual information about sensors to learn a well-aligned latent space between IMU and text. This facilitates mapping IMUs to textual activity representations in the subsequent stage. We compared ZeroHAR with 8 baselines on 5 benchmark HAR datasets to justify its efficacy on Zero-Shot HAR. Our ablations and case study highlight the superior alignment of IMU with text-based sensor metadata and activity representations. Aligning sensory data with relevant text metadata provides a platform to further investigate how such contextual knowledge can be leveraged for similar IoT-based applications. This will enable us to engage in more advanced natural language queries, reasoning, and responses related to sensory data.

Chapter 4, in part is currently being prepared for submission for publication of the material. Chowdhury, Ranak Roy; Kapila, Ritvik; Panse, Ameya; Zhang, Xiyuan; Teng, Diyan; Kulkarni, Rashmi; Hong, Dezhi; Gupta, Rajesh K.; Shang, Jingbo. The dissertation author was the primary investigator and the lead author of this paper.

Chapter 5

Conclusion and Future Works

This dissertation starts with a thesis that real-world sensory data are significantly short of labels. Hence, numerous sensor-based learning applications that rely on labeled data for supervised training can be improved by a) better utilizing the large corpus of unlabeled data, b) maximizing performance out of the limited labeled data, and c) transferring knowledge from the data for available classes to improve performance on novel, unseen classes.

We present a framework that addresses these challenges in order. Our framework first develops a pre-trained time-series model from unlabeled data by leveraging time interval information between consecutive samples to design time-sensitive contrastive learning and masked data reconstruction. This provides a warm initialization to the model parameters that excel even with limited labeled data. In the second stage, we design a fully-supervised multi-task finetuning for the supervised end task and an unsupervised task-aware data reconstruction that focuses on important timestamps, thereby improving end-task performance. Finally, in order to tackle classes that have no training data, we design a multi-modal framework that fuses text into sensor-based models to transfer knowledge from seen to unseen classes. Demonstrating broad applicability, our sensor time series analysis framework is effective in diverse sensing systems, ranging from small-scale individual healthcare monitoring, smart home automation, to large-scale smart building control, energy management, climate modeling and beyond.

5.1 Limitations and Open Challenges

The long-term objective of my research lies in designing data-efficient workflows for sensor-based learning systems. Below are a few potential directions that I am interested to explore in the near future.

Multi-modal Framework. Our current robust framework mostly targets only a single type of time series at a time. However, data collected from real-world sensing applications usually contain heterogeneous modalities. For example, a human activity recognition system might include multiple modalities such as IMU, EEG, microphone, and WiFi. If some domains collect data in different modalities, then the shortage of data in one modality may be compensated by the abundance of data in another modality.

Leveraging Large Language Models (LLMs). LLMs provide an automate way to generate rich source of information about different sensory domains. Leveraging external information banks, like LLMs, may compensate for the shortage of labels in many sensory applications. It will be important to engineer prompts that can generate relevant information about sensory domains aligned with real-world setting. We then need to discover ways in which this information can be integrated into existing sensory models to bolster performance.

Efficient Algorithms. Real-world sensor time series often feature high dimensions and extensive historical records. When paired with text metadata descriptions, the scale and complexity of the data further increase. Consequently, it is critical to design efficient processing algorithms, particularly for computations that occur on edge devices. Looking ahead, we plan to design quantized and pruned models for sensory applications and therefore improve the inference efficiency.

Real-Time Interactions. Our current framework transforms raw sensor time series into valuable insights to support downstream decision making processes. converts raw sensor time series into valuable insights that support downstream decision-making processes. However, users often provide feedback and ask additional questions after receiving these insights. For instance, in human activity recognition, users might offer real-time corrections for misclassified activities. This interaction presents an opportunity for the system to refine and improve its future predictions. Looking ahead, we intend to integrate user feedback directly into the system, enhancing real-time interactions between users and the framework.

Large-Scale Deployment. We focus not only on developing innovative framework designs but also on deploying and implementing these systems in real-world environments. Successfully running these robust learning methods necessitates both local and edge processing, as relying solely on cloud processing is impractical due to bandwidth limitations and privacy concerns. Effective edge processing demands careful partitioning of functions and restructuring of inference algorithms to allow the appropriate migration of computation and data. We plan to design FPGA co-processing methods to enable larger-scale deployment of our robust learning methods while maintaining efficiency.

Appendix A

Time-interval Aware Self-supervision

A.0.1 Notations

Notations used in Chapter 2 are detailed in Table A.1.

Table A.1. Notations used in Chapter 2.

Notation	Description
N	number of features
S	number of sampled timestamps
D	a single data point, $\mathrm{D} = (\mathrm{T}, \mathrm{X}, \mathrm{M})$, where $\mathrm{T} \in \mathbb{R}^{S}$ is sampling times, $\mathrm{X} \in \mathbb{R}^{S \times N}$ is feature vectors, and $\mathrm{M} \in \mathbb{R}^{S \times N}$ is binary mask
t, T_i	a particular sampling time
s	a particular feature
M_t	set of observed and unobserved features at timestamp t, $\mathrm{M}_t \in \mathbb{R}^{N}$
M_{tn}	whether feature n has been sampled at timestamp t or not
ω_{ih}, α_{ih}	learnable parameter for frequency and phase component of Time Embedding Layer
H	number of embedding functions
$\phi h(\mathrm{T})$	embedding function
TFA, FFA	Time-Feature and Feature-Feature/Self Attention Module
$\mathrm{Q}_T, \mathrm{K}_T$	Query and key vector for sampling times, T
$\mathrm{Q}_X, \mathrm{K}_X$	Query and key vector for features, X
Z	mean time interval computed from T
Z_i	mean time interval for timestamp T_i
$\mathrm{T}_{dense}, \mathrm{T}_{sparse}$	timestamps corresponding to the lowest and highest 50% values in Z
D_A	anchor augmented from D $\mathrm{D}_A = (\mathrm{T}_A, \mathrm{X}_{\mathrm{T}_A}, \mathrm{M}_{\mathrm{T}_A})$, where $\mathrm{T}_A, \mathrm{X}_{\mathrm{T}_A}, \mathrm{M}_{\mathrm{T}_A}$ are the sampling times, features and binary mask of the anchor
D_P	positive augmented from D $\mathrm{D}_P = (\mathrm{T}_P, \mathrm{X}_{\mathrm{T}_P}, \mathrm{M}_{\mathrm{T}_P})$, where $\mathrm{T}_P, \mathrm{X}_{\mathrm{T}_P}, \mathrm{M}_{\mathrm{T}_P}$ are the sampling times, features and binary mask of the positive
μ	fraction of T that will be sampled to form the anchor, remaining $1-\mu$ gets picked for the positive
μ_l, μ_u	lower and upper bound of μ during sampling
λ	fraction of T_{dense} to sample, remaining $1-\lambda$ gets picked from T_{sparse}
λ_l, λ_u	lower and upper bound of λ during sampling
B, τ	batch size, temperature parameter
J, α	number of masked segments and time interval to mask
q_n	timespan to mask for feature n
$\mathrm{X}_U, \mathrm{M}_U$	features and mask of the unmasked data
$\mathrm{X}_V, \mathrm{M}_V$	features and mask of the masked data
$\mathrm{X}_{V_n}[t : t + q_n], \mathrm{M}_{V_n}[t : t + q_n]$	target features and mask for the masked out data from t to $t + q_n$ for n
$\mathrm{X}_{U_n}[t : t + q_n], \mathrm{M}_{U_n}[t : t + q_n]$	features and mask for the unmasked data from t to $t + q_n$ for n
D_U, D_V	unmasked and masked data, $D_U = (\mathrm{T}, \mathrm{X}_U, \mathrm{M}_U)$, $D_V = (\mathrm{T}, \mathrm{X}_V, \mathrm{M}_V)$
$\tilde{\mathrm{X}}_U$	reconstruction of the masked out data
L, L_{Reco}, L_{CL}	Total Loss, Reconstruction Loss, Contrastive Loss
η	hyper-parameter to balance the contrastive and reconstruction loss

Appendix B

Task-specific Self-supervision

B.0.1 Notations

Notations used in Chapter 3 are detailed in Table B.1.

B.0.2 Environment

We have implemented TARNet in PYTHON with PYTORCH as the Deep Learning Library. All the experiments were conducted on a machine with AMD EPYC 7452 32-Core Processor @ 3.3GHz (2S/8C) and NVIDIA RTX A6000, running on Ubuntu 18.04.5 LTS (64-bit).

B.0.3 Training settings

We tuned TARNet-specific hyperparameters with a grid search on $\eta\{0.2, 0.5, 0.8\}$, $\mu\{0.05, 0.15, 0.25, 0.35\}$, $\beta\{0.4, 0.5, 0.6\}$, and $\lambda\{0.65, 0.80, 0.95\}$. Generally, $\eta = 0.5$, $\mu = 0.15$, $\beta = 0.5$, and $\lambda = 0.8$ are a good choice.

The best hyper-parameters for each dataset are reported in Table B.2 and Table B.3 for classification and regression experiments, respectively. Dropout and the number of training epochs are 0.01 and 300, respectively, across all datasets. We report the learning rate (lr), number of layers of transformer block (nlayer), hidden dimension of the model (nhid), number of self-attention heads (nhead), η, μ, β, and λ

Table B.1. Notations used in Chapter 3.

Notation	Description
S	sequence length
N	number of features
X	a single data point, $\mathrm{X} \in \mathbb{R}^{S\times N}$
y	target label
$\tilde{y}$	predicted label
p	probability distribution over C classes
x_t	feature vector at timestamp t
T_{END}	End Task
L_{END}	End Task Loss
T_{TAR}	Task-aware Reconstruction
L_{masked}	Task-aware Reconstruction Loss for the masked part of the data
$L_{unmasked}$	Task-aware Reconstruction Loss for the unmasked part of the data
λ	hyper-parameter to weight the masked, L_{masked}, and the unmasked loss, $L_{unmasked}$
L_{TAR}	Task-aware Reconstruction Loss
L_{Total}	Sum of end task loss, L_{END}, and task-aware reconstruction loss, L_{TAR}
η	hyper-parameter to weight the end task loss, L_{END}, and the task-aware reconstruction loss, L_{TAR}
C	Number of classes for classification
D	hidden dimension of the model
K_E	feed-forward dimension of Fully Connected Layers
W, b	all notations with any superscript or subscript on W and b refer to the learnable parameters of TARNet
m	binary mask, $m \in \mathbb{R}^S$
m_t	value of m at timestamp t
μ	masking ratio
$\tilde{\mathrm{X}}$	Reconstructed output of X
$\tilde{x_t}$	Reconstructed output of X at timestamp t, x_t
A	self-attention scores, $\mathrm{A} \in \mathbb{R}^{S\times S}$
A_{ik}	attention weight assigned to x_k during the update of x_i
σ	normalized aggregate attention weight assigned to all timestamps, $\sigma \in \mathbb{R}^S$
σ_k	normalized aggregate attention weight assigned to timestamp k
β	hyper-parameter to select important timestamps from σ
σ'	set of important timestamps chosen from σ

Table B.2. Best hyper-parameter settings for classification experiments.

Dataset	lr	nlayer	nhid	nhead	η	μ	β	λ
ArticularyWordRecognition	0.001	2	128	8	0.5	0.15	0.5	0.8
AtrialFibrillation	0.01	2	64	8	0.5	0.15	0.5	0.8
BasicMotions	0.01	2	64	8	0.5	0.15	0.5	0.8
CharacterTrajectories	0.0001	4	256	8	0.2	0.15	0.5	0.8
Cricket	0.001	1	64	8	0.5	0.05	0.5	0.8
DuckDuckGeese	0.0001	4	64	4	0.5	0.15	0.5	0.8
EigenWorms	0.01	1	64	8	0.5	0.15	0.5	0.8
Epilepsy	0.01	4	64	8	0.5	0.25	0.5	0.8
ERing	0.001	8	64	4	0.8	0.15	0.5	0.8
EthanolConcentration	0.0001	2	32	8	0.5	0.05	0.5	0.8
FaceDetection	0.001	2	256	8	0.2	0.25	0.5	0.8
FingerMovements	0.0001	2	128	8	0.5	0.15	0.5	0.8
HandMovementDirection	0.0001	2	64	8	0.8	0.15	0.5	0.8
Handwriting	0.0001	8	256	16	0.2	0.05	0.5	0.8
Heartbeat	0.001	1	64	8	0.5	0.15	0.5	0.8
InsectWingbeat	0.00001	8	64	8	0.2	0.05	0.5	0.8
JapaneseVowels	0.001	4	128	8	0.5	0.25	0.5	0.8
Libras	0.01	8	64	8	0.5	0.15	0.5	0.8
LSST	0.01	4	64	8	0.5	0.15	0.5	0.8
MotorImagery	0.001	1	64	8	0.2	0.15	0.5	0.8
NATOPS	0.0001	2	256	8	0.2	0.25	0.5	0.8
PEMS-SF	0.001	4	64	16	0.8	0.15	0.5	0.8
PenDigits	0.001	8	64	8	0.8	0.15	0.5	0.8
Phoneme	0.001	4	128	8	0.5	0.25	0.5	0.8
RacketSports	0.01	1	64	8	0.5	0.05	0.5	0.8
SelfRegulationSCP1	0.00001	2	64	8	0.5	0.15	0.5	0.8
SelfRegulationSCP2	0.0001	8	32	8	0.8	0.05	0.5	0.8
SpokenArabicDigits	0.001	2	128	8	0.5	0.35	0.5	0.8
StandWalkJump	0.001	4	64	8	0.5	0.35	0.5	0.8
UWaveGestureLibrary	0.001	8	64	8	0.2	0.15	0.5	0.8
PAMAP2	0.001	8	64	8	0.8	0.15	0.5	0.8
OpportunityGestures	0.001	8	64	8	0.2	0.25	0.5	0.8
OpportunityLocomotion	0.001	8	64	4	0.2	0.15	0.5	0.8
Occupancy	0.001	4	64	8	0.2	0.05	0.5	0.8

Table B.3. Best hyper-parameter settings for regression experiments.

Dataset	lr	nlayer	nhid	nhead	η	μ	β	λ
AppliancesEnergy	0.001	2	64	8	0.5	0.15	0.5	0.8
BenzeneConcentr.	0.001	2	64	8	0.5	0.15	0.5	0.8
BeijingPM10	0.01	2	64	8	0.5	0.15	0.5	0.8
BeijingPM25	0.01	2	64	8	0.5	0.15	0.5	0.8
LiveFuelMoisture	0.01	2	64	8	0.5	0.15	0.5	0.8
IEEEPPG	0.01	2	64	8	0.5	0.15	0.5	0.8

Appendix C

Sensor Context Aware Self-supervision

C.0.1 Notations

Notations used in Chapter 4 are detailed in Table C.1.

C.0.2 Experimental Details

Unseen class splits used in the test set are detailed in Table C.3. For each dataset, we run 3 experiments. Mathematically, for Experiment 1, 2, and 3, the test classes are, $U =$ fold 1, fold 2, and fold 3, respectively, while their corresponding training classes are, $O =$ fold 2 $\cup$ fold 3, fold 1 $\cup$ fold 3, and fold 1 $\cup$ fold 2, respectively.

In Table C.2, we measure ZeroHAR's relative percentage improvement over 2^{nd} best results (<u>underlined</u> values in Table 4.2 and 4.3). We average the Accuracy and Macro F1 across 5 datasets from Table 4.2 and 4.3. ZeroHAR achieves $+19.1\%$ and $+20.4\%$ relative improvement in Average Accuracy and Macro-F1, respectively, compared to the 2^{nd} best results.

Table C.1. Notations used in Chapter 4.

Notation	Description
O	set of observed activities
U	set of unobserved activities
G	$O \cup U$, set of all activities
a, g	given activity in G
N	Number of training data points
D_{tr}	Training set $= \{(X^i, Y^i) \| i = 1,2,3,...,N\}, Y^i \in O$
Z	Number of test data points
D_{te}	Test set $= \{(X^i, Y^i) \| i = 1,2,3,...,Z\}, Y^i \in U$
i	given data point
Y^i	label corresponding to X^i
B	set of body positions
b, e	given body position in B
M_b	set of IMU sensors at body position b
s, n	given IMU sensor
M_{sb}	IMU sensor s at body position b
W	$\{x, y, z\}$ axes
w, j	given axis in W
l	number of timestamps in data
X^i	IMU data for the i-the data point, $X^i \in \mathbb{R}^{\|W\| \Sigma_{b=1}^{\|B\|} \|M_b\| \times l}$
X^i_{wsb}	uni-variate time series from $w-$ axis of s from b, $X^i_{wsb} \in \mathbb{R}^l$
τ	temperature parameter in contrastive learning
t_{wsb}	"$w-$axis of s attached to b
h	hidden dimension of model
T_{wsb}	$R(ILM(t_{wsb})), T_{wsb} \in \mathbb{R}^h$
I^i_{wsb}	$P(K(X^i_{wsb}))I_{wsb}IN\mathbb{R}^h$
c	number of descriptions per activity, a
ϕ	Prompt
β	set of c descriptions for each of a activity, $\beta = LLM(\phi)$, where $\beta = \{\beta_1, \beta_2, ..., \beta_a, ..., \beta_{\|G\|}\}$, $\beta_a = \{\beta_{a_1}, \beta_{a_2}, ..., \beta_{a_c}\}$
I^i	$Q(K(X^i)), I^i \in \mathbb{R}^h$
α	$R(ILM(\beta)), \alpha \in \mathbb{R}^{c\|G\| \times h}$
A	Average of c embeddings per class from $\alpha, A \in \mathbb{R}^{\|G\| \times h}$
A^i	embedding for $Y^i, A^i \in \mathbb{R}^h$

Table C.2. Relative performance comparison of ZeroHAR with 2^{nd} best results.

	Average Accuracy	Average Macro F1
2^{nd} best results	50.28	37.24
ZeroHAR results	59.88	44.82
% relative improvement of ZeroHAR	+19.1%	+20.4%

Table C.3. Class splits across 3 folds.

Dataset	#Classes	folds
Opportunity [98]	11	**fold 1:** open door, close fridge, open dishwasher, close drawer. **fold 2:** close door, open fridge, close dishwasher, open drawer. **fold 3:** clean table, drink from cup, toggle switch.
PAMAP2 [97]	12	**fold 1:** nordic walking, descending stairs, ironing, rope jumping. **fold 2:** sitting, walking, ascending stairs, vacuum cleaning. **fold 3:** lying, standing, running, cycling.
Harth [77]	11	**fold 1:** cycling sit, cycling stand inactive, ascending stairs, running. **fold 2:** sitting, descending stairs, lying. **fold 3:** cycling sit inactive, standing, walking, shuffling.
USCHAD [145]	12	**fold 1:** walking forward, walking upstairs, jumping, sleeping. **fold 2:** walking left, walking downstairs, sitting, elevator up. **fold 3:** walking right, running forward, standing, elevator down.
WISDM [130]	18	**fold 1:** eating soup, walking, sitting, brushing teeth, kicking soccer ball, writing. **fold 2:** eating chips, jogging, standing, drinking from cup, playing catch with tennis ball, clapping. **fold 3:** eating pasta, stairs, typing, eating sandwich, dribbling basketball, folding clothes.

Bibliography

[1] Pulkit Agrawal, Ashvin V Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. *Advances in neural information processing systems*, 29, 2016.

[2] Fadi Al Machot, Mohammed R. Elkobaisi, and Kyandoghere Kyamakya. Zero-shot human activity recognition using non-visual sensors. *Sensors*, 20(3), 2020.

[3] D. Alleysson, S. Süsstrunk, and J. Hérault. Linear demosaicing inspired by the human visual system. *IEEE Transactions on Image Processing*, 14:439–449, 2005.

[4] Berna Altinel and M. Ganiz. Semantic text classification: A survey of past and recent advances. *Inf. Process. Manag.*, 54:1129–1153, 2018.

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

[6] Anthony Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn Keogh. The uea multivariate time series classification archive, 2018. *arXiv preprint arXiv:1811.00075*, 2018.

[7] Yue Bai, Lichen Wang, Zhiqiang Tao, Sheng Li, and Yun Fu. Correlative channel-aware fusion for multi-view time series classification. In *AAAI*, number 8, 2021.

[8] Nicolas Barascud, M. Pearce, T. Griffiths, Karl J. Friston, and M. Chait. Brain responses in humans reveal ideal observer-like sensitivity to complex acoustic patterns. *Proceedings of the National Academy of Sciences*, 113:E616 – E625, 2016.

[9] Mustafa Gokce Baydogan and George Runger. Learning a symbolic representation for multivariate time series classification. *Data Mining and Knowledge Discovery*, 29(2):400–422, 2015.

[10] Sathvik Bhaskarpandit, Priyanka Gupta, and Manik Gupta. Lets-gzsl: A latent embedding model for time series generalized zero shot learning. *arXiv preprint arXiv:2207.12007*, 2022.

[11] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[12] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1):1–12, 2018.

[13] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[14] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *KDD*, pages 785–794, 2016.

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

[16] Xiaokang Chen, Mingyu Ding, Xiaodi Wang, Ying Xin, Shentong Mo, Yunhao Wang, Shumin Han, Ping Luo, Gang Zeng, and Jingdong Wang. Context autoencoder for self-supervised representation learning. *International Journal of Computer Vision*, 132(1):208–223, 2024.

[17] Zhenghua Chen, Le Zhang, Chaoyang Jiang, Zhiguang Cao, and Wei Cui. Wifi csi based passive human activity recognition using attention based blstm. *IEEE Transactions on Mobile Computing*, 18:2714–2724, 2019.

[18] Heng-Tze Cheng, Martin Griss, Paul Davis, Jianguo Li, and Di You. Towards zero-shot learning for human activity recognition using semantic attribute sequence model. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 355–358, 2013.

[19] Heng-Tze Cheng, Feng-Tso Sun, Martin Griss, Paul Davis, Jianguo Li, and Di You. Nuactiv: Recognizing unseen new activities using semantic attribute-based learning. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, pages 361–374, 2013.

[20] Hyeongju Choi, Apoorva Beedu, and Irfan Essa. Multimodal contrastive learning with hard negative sampling for human activity recognition. *arXiv preprint arXiv:2309.01262*, 2023.

[21] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

[22] Ranak Roy Chowdhury, Muhammad Abdullah Adnan, and Rajesh K Gupta. Real-time principal component analysis. *ACM Transactions on Data Science*, 1(2):1–36, 2020.

[23] Ranak Roy Chowdhury, Jiacheng Li, Xiyuan Zhang, Dezhi Hong, Rajesh K Gupta, and Jingbo Shang. Primenet: Pre-training for irregular multivariate time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 7184–7192, 2023.

[24] Ranak Roy Chowdhury, Xiyuan Zhang, Jingbo Shang, Rajesh K Gupta, and Dezhi Hong. Tarnet: Task-aware reconstruction for time-series transformer. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA*, pages 14–18, 2022.

[25] Federico Cruciani, Anastasios Vafeiadis, Chris Nugent, Ian Cleland, Paul McCullagh, Konstantinos Votis, Dimitrios Giakoumis, Dimitrios Tzovaras, Liming Chen, and Raouf Hamzaoui. Feature learning for human activity recognition using convolutional neural networks: A case study for inertial measurement unit and audio data. *CCF Transactions on Pervasive Computing and Interaction*, 2(1):18–32, 2020.

[26] Pathirage N Deelaka, Devin Y De Silva, Sandareka Wickramanayake, Dulani Meedeniya, and Sanka Rasnayaka. Tezarnet: Temporal zero-shot activity recognition network. In *International Conference on Neural Information Processing*, pages 444–455. Springer, 2023.

[27] Angus Dempster, François Petitjean, and Geoffrey I Webb. Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, 34(5):1454–1495, 2020.

[28] Angus Dempster, Daniel F Schmidt, and Geoffrey I Webb. Minirocket: A very fast (almost) deterministic transform for time series classification. In *KDD*, 2021.

[29] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[30] A. Doherty, Steve Hodges, A. King, A. Smeaton, E. Berry, C. Moulin, Siân E. Lindley, P. Kelly, and C. Foster. Wearable cameras in health: the state of the art and future possibilities. *American journal of preventive medicine*, 44 3:320–3, 2013.

[31] Abhilash Dorle, Fangyu Li, Wenzhan Song, and Sheng Li. Learning discriminative virtual sequences for time series classification. In *CIKM*, pages 2001–2004, 2020.

[32] Harris Drucker, Chris JC Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. Support vector regression machines. *NIPS*, 9:155–161, 1997.

[33] Shengdong Du, Tianrui Li, Yan Yang, and S. Horng. Multivariate time series forecasting via attention-based encoder-decoder framework. *Neurocomputing*, 388:269–279, 2020.

[34] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

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

[36] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4):917–963, 2019.

[37] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F Schmidt, Jonathan Weber, Geoffrey I Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery*, 34(6), 2020.

[38] Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. Unsupervised scalable representation learning for multivariate time series. *arXiv preprint arXiv:1901.10738*, 2019.

[39] Tak-Chung Fu. A review on time series data mining. *Eng. Appl. Artif. Intell.*, 24:164–181, 2011.

[40] Qinhua Gao, Jie Wang, Xiaorui Ma, Xueyan Feng, and Hongyu Wang. Csi-based device-free wireless localization and activity recognition using radio image features. *IEEE Transactions on Vehicular Technology*, 66:10346–10356, 2017.

[41] S James Gates Jr, Yangrui Hu, and S-N Hazel Mak. Advening to adynkrafields: Young tableaux to component fields of the 10d, n= 1 scalar superfield. *arXiv preprint arXiv:2006.03609*, 2020.

[42] Rohit Girdhar, Alaaeldin El-Nouby, Zhuang Liu, Mannat Singh, Kalyan Vasudev Alwala, Armand Joulin, and Ishan Misra. Imagebind: One embedding space to bind them all. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15180–15190, 2023.

[43] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 270–279, 2017.

[44] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

[45] Gauri Gupta, Ritvik Kapila, Keshav Gupta, and Ramesh Raskar. Domain generalization in robust invariant representation. *arXiv preprint arXiv:2304.03431*, 2023.

[46] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

[47] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[48] William Hogan, Jiacheng Li, and Jingbo Shang. Fine-grained contrastive learning for relation extraction. *ArXiv*, abs/2205.12491, 2022.

[49] Dezhi Hong, Jorge Ortiz, Kamin Whitehouse, and David Culler. Towards automatic spatial verification of sensor placement in buildings. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*, pages 1–8, 2013.

[50] Max Horn, Michael Moor, Christian Bock, Bastian Rieck, and Karsten Borgwardt. Set functions for time series. In *International Conference on Machine Learning*, pages 4353–4363. PMLR, 2020.

[51] Lu Hou, James Kwok, and Jacek Zurada. Efficient learning of timeseries shapelets. In *AAAI*, volume 30, 2016.

[52] Tsung-Yu Hsieh, Suhang Wang, Yiwei Sun, and Vasant Honavar. Explainable multivariate time series classification: A deep neural network which learns to attend to important variables as well as time intervals. In *WSDM*, 2021.

[53] Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM transactions on audio, speech, and language processing*, 29:3451–3460, 2021.

[54] Zawar Hussain, Michael Sheng, and Wei Emma Zhang. Different approaches for human activity recognition: A survey. *arXiv preprint arXiv:1906.05074*, 2019.

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

[56] Wenchao Jiang and Zhaozheng Yin. Human activity recognition using wearable sensors by deep convolutional neural networks. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 1307–1310, 2015.

[57] Charmi Jobanputra, Jatna Bavishi, and Nishant Doshi. Human activity recognition: A survey. *Procedia Computer Science*, 155:698–703, 2019.

[58] Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016.

[59] Boštjan Kaluža, Violeta Mirchevska, Erik Dovgan, Mitja Luštrek, and Matjaž Gams. An agent-based approach to care in independent living. In *International joint conference on ambient intelligence*, pages 177–186. Springer, 2010.

[60] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Samuel Harford. Multivariate lstm-fcns for time series classification. *Neural Networks*, 116:237–245, 2019.

[61] Seyed Mehran Kazemi, Rishab Goel, Sepehr Eghbali, Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, Pascal Poupart, and Marcus Brubaker. Time2vec: Learning a vector representation of time. *arXiv preprint arXiv:1907.05321*, 2019.

[62] D. Keim. Designing pixel-oriented visualization techniques: Theory and applications. *IEEE Trans. Vis. Comput. Graph.*, 6:59–78, 2000.

[63] P. Kelly, S. Marshall, H. Badland, J. Kerr, M. Oliver, A. Doherty, and C. Foster. An ethical framework for automated, wearable cameras in health behavior research. *American journal of preventive medicine*, 44 3:314–9, 2013.

[64] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019.

[65] Bulat Khaertdinov and Stylianos Asteriadis. Temporal feature alignment in contrastive self-supervised learning for human activity recognition. In *2022 IEEE International Joint Conference on Biometrics (IJCB)*, pages 1–9. IEEE, 2022.

[66] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[67] Tejas D Kulkarni, Ankush Gupta, Catalin Ionescu, Sebastian Borgeaud, Malcolm Reynolds, Andrew Zisserman, and Volodymyr Mnih. Unsupervised learning of object keypoints for perception and control. *Advances in neural information processing systems*, 32, 2019.

[68] Oscar D Lara and Miguel A Labrador. A survey on human activity recognition using wearable sensors. *IEEE communications surveys & tutorials*, 15(3):1192–1209, 2012.

[69] Mathias Lechner and Ramin Hasani. Learning long-term dependencies in irregularly-sampled time series. *arXiv preprint arXiv:2006.04418*, 2020.

[70] Dongha Lee, Seonghyeon Lee, and Hwanjo Yu. Learnable dynamic temporal pooling for time series classification. *arXiv preprint arXiv:2104.02577*, 2021.

[71] Guozhong Li, Byron Choi, Jianliang Xu, Sourav S Bhowmick, Kwok-Pan Chun, and Grace LH Wong. Shapenet: A shapelet-neural network approach for multivariate time series classification. In *AAAI*, volume 35, pages 8375–8383, 2021.

[72] Jiacheng Li, Jingbo Shang, and Julian McAuley. UCTopic: Unsupervised contrastive learning for phrase representations and topic mining. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6159–6169, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[73] Shuheng Li, Ranak Roy Chowdhury, Jingbo Shang, Rajesh K Gupta, and Dezhi Hong. Units: Short-time fourier inspired neural networks for sensory time series classification. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, pages 234–247, 2021.

[74] Haoran Liang, Lei Song, Jianxing Wang, Lili Guo, Xuzhi Li, and Ji Liang. Robust unsupervised anomaly detection via multi-time scale dcgans with forgetting mechanism for industrial multivariate time series. *Neurocomputing*, 423:444–462, 2021.

[75] Shichen Liu, Mingsheng Long, Jianmin Wang, and Michael I Jordan. Generalized zero-shot learning with deep calibration network. *Advances in neural information processing systems*, 31, 2018.

[76] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[77] Aleksej Logacjov, Kerstin Bach, Atle Kongsvold, Hilde Bremseth Bårdstu, and Paul Jarle Mork. Harth: A human activity recognition dataset for machine learning. *Sensors*, 21(23):7853, 2021.

[78] Qianli Ma, Zhenjing Zheng, Jiawei Zheng, Sen Li, Wanqing Zhuang, and Garrison W Cottrell. Joint-label learning by dual augmentation for time series classification. In *AAAI*, volume 35, pages 8847–8855, 2021.

[79] Qianli Ma, Wanqing Zhuang, Sen Li, Desen Huang, and Garrison Cottrell. Adversarial dynamic shapelet networks. In *AAAI*, volume 34, pages 5069–5076, 2020.

[80] Benjamin M Marlin, David C Kale, Robinder G Khemani, and Randall C Wetzel. Unsupervised pattern discovery in electronic health care data using probabilistic clustering models. In *Proceedings of the 2nd ACM SIGHIT international health informatics symposium*, pages 389–398, 2012.

[81] Moe Matsuki, Paula Lago, and Sozo Inoue. Characterizing word embeddings for zero-shot sensor-based human activity recognition. *Sensors*, 19(22):5043, 2019.

[82] Josh H. McDermott, Michael Schemitsch, and Eero P. Simoncelli. Summary statistics in auditory perception. *Nature Neuroscience*, 16:493–498, 2013.

[83] A. Merlo, D. Farina, and R. Merletti. A fast and reliable technique for muscle activity detection from surface emg signals. *IEEE Transactions on Biomedical Engineering*, 50:316–323, 2003.

[84] Hazar Mliki, Fatma Bouhlel, and Mohamed Hammami. Human activity recognition from uav-captured video sequences. *Pattern Recognit.*, 100:107140, 2020.

[85] Masoud Mohtadifar, M. Cheffena, and Alireza Pourafzal. Acoustic- and radio-frequency-based human activity recognition. *Sensors (Basel, Switzerland)*, 22, 2022.

[86] Seungwhan Moon, Andrea Madotto, Zhaojiang Lin, Alireza Dirafzoon, Aparajita Saraf, Amy Bearman, and Babak Damavandi. Imu2clip: Multimodal contrastive learning for imu motion sensors from egocentric videos and text. *arXiv preprint arXiv:2210.14395*, 2022.

[87] Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. Phased lstm: Accelerating recurrent network training for long or event-based sequences. *Advances in neural information processing systems*, 29, 2016.

[88] Stavros Ntalampiras and Ilyas Potamitis. Transfer learning for improved audio-based human activity recognition. *Biosensors*, 8(3):60, 2018.

[89] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[90] OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023.

[91] Hao Peng, Jianxin Li, Qiran Gong, Senzhang Wang, Lifang He, Bo Li, Lihong Wang, and Philip S. Yu. Hierarchical taxonomy-aware and attentional graph capsule rcnns for large-scale multi-label text classification. *IEEE Transactions on Knowledge and Data Engineering*, 33:2505–2519, 2019.

[92] Trang Pham, Truyen Tran, Dinh Phung, and Svetha Venkatesh. Predicting healthcare trajectories from medical records: A deep learning approach. *Journal of biomedical informatics*, 69:218–229, 2017.

[93] Matt Posik and Bernd Surrow. Construction of triple-gem detectors using commercially manufactured large gem foils. In *2016 IEEE Nuclear Science Symposium, Medical Imaging Conference and Room-Temperature Semiconductor Detector Workshop (NSS/MIC/RTSD)*, pages 1–5. IEEE, 2016.

[94] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[95] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[96] Roshan Rao, Nicholas Bhattacharya, Neil Thomas, Yan Duan, Peter Chen, John Canny, Pieter Abbeel, and Yun Song. Evaluating protein transfer learning with tape. *Advances in neural information processing systems*, 32, 2019.

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

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

[99] M. Rostaghi and H. Azami. Dispersion entropy: A measure for time-series analysis. *IEEE Signal Processing Letters*, 23:610–614, 2016.

[100] Yulia Rubanova, Ricky TQ Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. *Advances in neural information processing systems*, 32, 2019.

[101] Patrick Schäfer and Ulf Leser. Multivariate time series classification with weasel+ muse. *arXiv preprint arXiv:1711.11343*, 2017.

[102] Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. wav2vec: Unsupervised pre-training for speech recognition. *arXiv preprint arXiv:1904.05862*, 2019.

[103] Mohammad Shokoohi-Yekta, Jun Wang, and Eamonn Keogh. On the non-trivial generalization of dynamic time warping to the multi-dimensional case. In *ICDM*, pages 289–297. SIAM, 2015.

[104] Satya Narayan Shukla and Benjamin M Marlin. Interpolation-prediction networks for irregularly sampled time series. *arXiv preprint arXiv:1909.07782*, 2019.

[105] Satya Narayan Shukla and Benjamin M Marlin. Multi-time attention networks for irregularly sampled time series. *arXiv preprint arXiv:2101.10318*, 2021.

[106] Ikaro Silva, George Moody, Daniel J Scott, Leo A Celi, and Roger G Mark. Predicting in-hospital mortality of icu patients: The physionet/computing in cardiology challenge 2012. In *2012 Computing in Cardiology*, pages 245–248. IEEE, 2012.

[107] M. Simão, N. Mendes, O. Gibaru, and P. Neto. A review on electromyography decoding and pattern recognition for human-machine interaction. *IEEE Access*, 7:39564–39582, 2019.

[108] Tej Singh and Dinesh Kumar Vishwakarma. A deeply coupled convnet for human activity recognition using dynamic and rgb images. *Neural Computing and Applications*, 33(1):469–485, 2021.

[109] Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. Zero-shot learning through cross-modal transfer. *Advances in neural information processing systems*, 26, 2013.

[110] Zhi-Da Song, Luis Elcoro, and B Andrei Bernevig. Twisted bulk-boundary correspondence of fragile topology. *Science*, 367(6479):794–797, 2020.

[111] J. A. Stork, Luciano Spinello, Jens Silva, and K. Arras. Audio-based human activity recognition using non-markovian ensemble voting. *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*, pages 509–514, 2012.

[112] Vladimir Svetnik, Andy Liaw, Christopher Tong, J Christopher Culberson, Robert P Sheridan, and Bradley P Feuston. Random forest: a classification and regression tool for compound classification and qsar modeling. *Journal of chemical information and computer sciences*, 43(6):1947–1958, 2003.

[113] Nima Tajbakhsh, Jae Y Shin, Suryakanth R Gurudu, R Todd Hurst, Christopher B Kendall, Michael B Gotway, and Jianming Liang. Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE transactions on medical imaging*, 35(5):1299–1312, 2016.

[114] Chang Wei Tan, Christoph Bergmeir, Francois Petitjean, and Geoffrey I Webb. Monash university, uea, ucr time series regression archive. *arXiv e-prints*, pages arXiv–2006, 2020.

[115] Chang Wei Tan, Christoph Bergmeir, Francois Petitjean, and Geoffrey I Webb. Time series extrinsic regression. *Data Mining and Knowledge Discovery*, pages 1–29, 2021.

[116] Qingxiong Tan, Mang Ye, Baoyao Yang, Siqi Liu, Andy Jinhua Ma, Terry Cheuk-Fung Yip, Grace Lai-Hung Wong, and PongChi Yuen. Data-gru: Dual-attention time-aware gated recurrent unit for irregular multivariate time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 930–937, 2020.

[117] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*, pages 776–794. Springer, 2020.

[118] Sana Tonekaboni, Danny Eytan, and Anna Goldenberg. Unsupervised representation learning for time series with temporal neighborhood coding. *arXiv preprint arXiv:2106.00750*, 2021.

[119] Catherine Tong, Jinchen Ge, and Nicholas D Lane. Zero-shot learning for imu-based activity recognition using video embeddings. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 5(4):1–23, 2021.

[120] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[121] Hadiqa Aman Ullah, S. Letchmunan, M. S. Zia, Umair Muneer Butt, and F. H. Hassan. Analysis of deep neural networks for human activity recognition in videos—a systematic literature review. *IEEE Access*, 9:126366–126387, 2021.

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

[123] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *arXiv preprint arXiv:1809.10341*, 2018.

[124] Dinesh Kumar Vishwakarma and Kuldeep Singh. Human activity recognition based on spatial distribution of gradients at sublevels of average energy silhouette images. *IEEE Transactions on Cognitive and Developmental Systems*, 9(4):316–327, 2016.

[125] Michalis Vrigkas, Christophoros Nikou, and Ioannis A Kakadiaris. A review of human activity recognition methods. *Frontiers in Robotics and AI*, 2:28, 2015.

[126] Wei Wang, Chunyan Miao, and Shuji Hao. Zero-shot human activity recognition via nonlinear compatibility based method. In *Proceedings of the International Conference on Web Intelligence*, pages 322–330, 2017.

[127] Wen Wang, A. Liu, Muhammad Shahzad, Kang Ling, and Sanglu Lu. Device-free human activity recognition using commercial wifi devices. *IEEE Journal on Selected Areas in Communications*, 35:1118–1131, 2017.

[128] Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, pages 1578–1585. IEEE, 2017.

[129] Philip B. Weerakody, K. K. F. Wong, Guanjin Wang, and W. Ela. A review of irregular time series data handling with gated recurrent neural networks. *Neurocomputing*, 441:161–178, 2021.

[130] Gary M Weiss, Kenichi Yoneda, and Thaier Hayajneh. Smartphone and smartwatch-based biometrics using activities of daily living. *IEEE Access*, 7:133190–133202, 2019.

[131] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. Transformers in time series: A survey. *arXiv preprint arXiv:2202.07125*, 2022.

[132] Xinshuo Weng and Kris Kitani. Learning spatio-temporal features with two-stream deep 3d cnns for lipreading. *arXiv preprint arXiv:1905.02540*, 2019.

[133] Tong Wu, Yiqiang Chen, Yang Gu, Jiwei Wang, Siyu Zhang, and Zhanghu Zhechen. Multi-layer cross loss model for zero-shot human activity recognition. In *Advances in Knowledge Discovery and Data Mining: 24th Pacific-Asia Conference, PAKDD 2020, Singapore, May 11–14, 2020, Proceedings, Part I 24*, pages 210–221. Springer, 2020.

[134] Xiaopeng Xi, Eamonn Keogh, Christian Shelton, Li Wei, and Chotirat Ann Ratanamahatana. Fast time series classification using numerosity reduction. In *ICML*, pages 1033–1040, 2006.

[135] Yongqin Xian, Christoph H Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE transactions on pattern analysis and machine intelligence*, 41(9):2251–2265, 2018.

[136] Xukai Xie, Yuan Zhou, and Sun-Yuan Kung. Hgc: Hierarchical group convolution for highly efficient neural network. *arXiv preprint arXiv:1906.03657*, 2019.

[137] Chao-Han Huck Yang, Yun-Yun Tsai, and Pin-Yu Chen. Voice2series: Reprogramming acoustic models for time series classification. In *ICML*, 2021.

[138] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

[139] Lexiang Ye and Eamonn Keogh. Time series shapelets: a new primitive for data mining. In *KDD*, pages 947–956, 2009.

[140] Siamak Yousefi, Hirokazu Narui, Sankalp Dayal, Stefano Ermon, and S. Valaee. A survey on behavior recognition using wifi channel state information. *IEEE Communications Magazine*, 55:98–104, 2017.

[141] Ye Yuan, Guangxu Xun, Fenglong Ma, Yaqing Wang, Nan Du, Kebin Jia, Lu Su, and Aidong Zhang. Muvan: A multi-view attention network for multivariate temporal data. In *ICDM*, pages 717–726. IEEE, 2018.

[142] Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. Ts2vec: Towards universal representation of time series. *arXiv preprint arXiv:2106.10466*, 2021.

[143] Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, and Bixiong Xu. Learning timestamp-level representations for time series with hierarchical contrastive loss. *arXiv preprint arXiv:2106.10466*, 2021.

[144] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. A transformer-based framework for multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2114–2124, 2021.

[145] Mi Zhang and Alexander A Sawchuk. Usc-had: A daily activity dataset for ubiquitous activity recognition using wearable sensors. In *Proceedings of the 2012 ACM conference on ubiquitous computing*, pages 1036–1043, 2012.

[146] Xiyuan Zhang, Ranak Roy Chowdhury, Dezhi Hong, Rajesh K Gupta, and Jingbo Shang. Modeling label semantics improves activity recognition. *arXiv preprint arXiv:2301.03462*, 2023.

[147] Xiyuan Zhang, Ranak Roy Chowdhury, Jingbo Shang, Rajesh Gupta, and Dezhi Hong. Esc-gan: Extending spatial coverage of physical sensors. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 1347–1356, 2022.

[148] Xuchao Zhang, Yifeng Gao, Jessica Lin, and Chang-Tien Lu. Tapnet: Multivariate time series classification with attentional prototypical network. In *AAAI*, number 04, 2020.

[149] Yuan Zhang. Attain: Attention-based time-aware lstm networks for disease progression modeling. In *In Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI-2019), pp. 4369-4375, Macao, China.*, 2019.

[150] Yi Zheng, Qi Liu, Enhong Chen, Yong Ge, and J Leon Zhao. Time series classification using multi-channels deep convolutional neural networks. In *International conference on web-age information management*, pages 298–310. Springer, 2014.

[151] Rui Zhou, Running Zhao, and Edith CH Ngai. Human activity recognition from motion and acoustic sensors using contrastive learning. In *2023 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW)*, pages 1–4. IEEE, 2023.

[152] Yunjiao Zhou, Jianfei Yang, Han Zou, and Lihua Xie. Tent: Connect language models with iot sensors for zero-shot activity recognition. *arXiv preprint arXiv:2311.08245*, 2023.