Self-supervised learning for data-efficient human activity recognition

Chi Ian Tang

Hughes Hall
University of Cambridge

November 2023

This thesis is submitted for the degree of Doctor of Philosophy
at the Department of Computer Science and Technology

Declaration

This thesis is the result of my own work and includes nothing which is the outcome of work done in collaboration except as declared in the Preface and specified in the text. I further state that no substantial part of my thesis has already been submitted, or, is being concurrently submitted for any such degree, diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text. It does not exceed the prescribed word limit for the relevant Degree Committee.

Chi Ian Tang
November 2023

Self-supervised learning for data-efficient human activity recognition

Chi Ian Tang

Summary

Over the last decade, smart mobile devices have become ubiquitous, bringing about significant lifestyle changes worldwide. Mobile sensing, which involves obtaining and analysing data from mobile devices and the environment, has emerged as an active research area. It captures the unique opportunity for mobile devices to offer insight into user behaviours. Within mobile sensing, human activity recognition is a fundamental task that aims to identify users' physical actions. Motivated by advancements in deep learning, human activity recognition research has also widely adopted these methods. However, compared to other data modalities, human activity recognition models struggle with the limited availability of labels, due to the difficulty of ground-truth collection. These models often fail to generalise across different users, devices, and changing data distributions. This thesis tackles these challenges by developing and evaluating novel training paradigms. Our proposed paradigms leverage data from additional sources, including other devices and readily available unlabelled data that can be collected easily and often passively, to provide supervision for deep learning, enabling human activity recognition models to be more data-efficient.

First, we proposed a new semi-supervised training pipeline that combines self-supervised learning and knowledge distillation to effectively leverage large-scale unlabelled datasets for human activity recognition. This helps models generalise better across different users by increasing the diversity of data that the model is trained on through augmentation and unlabelled data.

Next, we designed a collaborative self-supervised learning technique that leverages unlabelled data from multiple devices carried by a user. This method is inspired by the insight that data from multiple devices capture the same physical activity from different viewpoints. The contrastive learning setup, which makes representations for samples from different devices to be similar, is used to extract high-quality features from the data.

Finally, we developed continual learning methods motivated by observations that user behaviour often shifts over time due to lifestyle changes. These methods help models better adapt to changing data distributions and learn from new data. We first proposed a multi-task training method that allows models to have better flexibility in adapting to new tasks. Then, we developed a continual learning strategy that balances retaining prior knowledge and learning from new data. This strategy uses self-supervised learning for knowledge retention and a carefully designed loss function to balance different learning objectives.

Through extensive evaluation on open datasets, the training paradigms proposed in this thesis provide evidence for and contribute to the development of data-efficient human activity recognition systems by leveraging readily-available data through self-supervised learning.

Acknowledgements

I am incredibly lucky and privileged to have the opportunity to pursue a PhD, and this thesis represents not only my own work, but also the care, guidance, assistance, and wisdom given to me by many others. To all of you who have been with me on this journey – I am very grateful to have you, and this journey would not be the same without you.

First, I would like to express my deepest appreciation and gratitude to Cecilia Mascolo, my advisor. Her unwavering support has been absolutely instrumental in the completion of this thesis. From the first chat we had about pursuing a PhD during my master's degree, the time I was lost in my research direction, throughout the COVID times when we had to switch to online meetings, to the final stages of thesis writing, she has been supportive and accommodating to my personal learning process. Her ability to identify meaningful research problems and her insistence on conceptual clarity pushed my work to have a stronger impact. The Mobile Systems Group at the University of Cambridge led by Cecilia has been an academically strong and welcoming community, which I am proud to be part of.

I am also extremely fortunate to work with brilliant minds including my colleagues, co-authors, and mentors. At a time when all of us were still adapting to life during the pandemic, I am grateful to have met Akhil Mathur, who has been a fantastic mentor, collaborator and friend. He allowed me to pick up the pace again and introduced me to a wider research community beyond mobile computing. The collaborations with him have played an important role in shaping this thesis. I would also like to express my appreciation for the support and guidance that I received from Fahim Kawsar. Through him, I am glad to be part of the Nokia Bell Labs family, and many collaborations would not have happened without his support. And I cannot forget to mention Dimitris Spathis, who has been an amazing friend and colleague of mine. The discussions with him at the beginning of my PhD allowed me to orient myself in the daunting task of defining my own research questions. I am glad that we have had many fruitful collaborations, and I appreciate his openness and kindness as a friend.

In addition, I would like to express my gratitude to my examiners, Professors Cengiz Öztireli and Yu Guan, for dedicating their time and effort to going through this thesis. I am appreciative of their patience, and their feedback has certainly improved this thesis.

It is also important to mention that my college, Hughes Hall, has been my second home during my PhD. I am glad that I had the opportunity to contribute to the student community by being on the MCR committee for four years. During this time, I had the chance to meet people from different backgrounds, work with different committees, and make lifelong friendships. All the fun times, memories, laughter and companionship connected to Hughes Hall will definitely last a lifetime.

Furthermore, I am immensely grateful for the financial support from the Cambridge Trust, the Nokia Corporation, and the Doris Zimmern Charitable Foundation. Their generous funding allowed me to pursue my dream fearlessly and embark on this life-changing journey. The opportunity they provided is sincerely appreciated.

Last, but most importantly, I would like to dedicate this thesis to my beloved parents and sister. Having the opportunity to pursue a PhD does not come easy, and this would not have been possible without their endless love, support, and confidence in me. I am forever grateful for their support for whatever I want to pursue, and for having me in their hearts all the time. This solid foundation of love has undoubtedly been supporting me through the ups and downs of this journey, and I would like to mention that I love you all dearly from the bottom of my heart.

The path to completing my doctoral degree has been long and challenging, but I am grateful to have many people who support me along the way. I cannot express enough my sincere gratitude to my advisor, committee, colleagues, friends, and family. I hope that I will have the chance to pay it forward and make a positive impact through my research.

Contents

1 Introduction **15**

1.1 Motivation . . . 16

1.2 Thesis and substantiation . . . 17

1.3 Contributions and chapter outline . . . 18

1.4 List of publications . . . 21

2 Background **25**

2.1 Human activity recognition . . . 25

2.1.1 Sensor-based HAR . . . 26

2.1.2 Data acquisition . . . 26

2.1.3 Modelling sensor time-series . . . 27

2.2 Training paradigms . . . 32

2.2.1 Semi-supervised learning . . . 33

2.2.2 Self-supervised learning . . . 33

2.2.3 Contrastive learning . . . 37

2.2.4 Self-training . . . 38

2.2.5 Transfer learning . . . 39

2.2.6 Multi-task learning . . . 40

2.2.7 Continual learning . . . 40

2.3 Research gap . . . 42

3 Improving human activity recognition through self-training with unlabelled data **45**

3.1 Motivation and overview . . . 45

3.2 Approach . . . 47

3.2.1 Problem statement . . . 47

3.2.2 The SelfHAR training pipeline . . . 47

3.2.3 Configurations of the SelfHAR pipeline . . . 52

3.3 Experimental protocols . . . 54

3.3.1 Datasets . . . 54

3.3.2 Data preparation 59
3.3.3 Experimental setup 59
3.4 Evaluation and discussion 61
3.4.1 Comparison against baseline algorithms 61
3.4.2 Ablation studies: effectiveness of the combined pre-training tasks . 65
3.4.3 Impact of training with limited labelled data 68
3.4.4 Visualisation of extracted features using t-SNE 70
3.4.5 Evaluating the importance of using unlabelled data 71
3.4.6 Leveraging large unlabelled datasets to improve supervised performance: PAAS and Fenland case study 73
3.5 Conclusions 74

4 Collaborative contrastive learning for human activity recognition 75
4.1 Motivation and overview 75
4.2 Time-synchronous multi-device systems 77
4.2.1 Concepts 77
4.2.2 Problem statement 78
4.2.3 Research challenges 79
4.3 Approach 81
4.3.1 Solution overview 82
4.3.2 Feature extractor 82
4.3.3 Device selection and contrastive sampling 83
4.3.4 Multi-view contrastive loss 86
4.4 Experimental protocols 87
4.4.1 Datasets 87
4.4.2 Baselines 89
4.4.3 Data preparation 91
4.4.4 Experimental setup 91
4.5 Evaluation and discussion 92
4.5.1 Performance in the low-data regime 92
4.5.2 Comparison of recognition performance with baselines 94
4.5.3 Embedding similarity and data saliency 96
4.5.4 Analysis of the device selection strategy 99
4.5.5 Analysis of the contrastive sampling approach 99
4.5.6 Analysing the role of weights in the multi-view contrastive loss . . . 100
4.5.7 Robustness to sensor heterogeneity 101
4.5.8 Robustness to missing devices 102
4.5.9 Robustness to temporal misalignment 103
4.5.10 Generalisability of the feature extractor 103

4.5.11 Discussion and limitations 104
4.6 Conclusions 105

5 Overcoming catastrophic forgetting with multi-task and self-supervised learning 107
5.1 Motivation and overview 107
5.2 Improving feature generalisability with multi-task learning in class incremental learning 108
5.2.1 Approach 109
5.2.2 Experimental protocols 111
5.2.3 Evaluation and discussion 114
5.3 Practical self-supervised continual learning with continual fine-tuning . . . 119
5.3.1 Approach 121
5.3.2 Experimental protocols 124
5.3.3 Evaluation and discussion 126
5.4 Conclusions 138

6 Conclusion 139
6.1 Summary of contributions 139
6.1.1 Improving human activity recognition through self-training with unlabelled data 139
6.1.2 Collaborative contrastive learning for human activity recognition . . 140
6.1.3 Overcoming catastrophic forgetting with multi-task and self-supervised learning 140
6.2 Implications and limitations 141
6.3 Future research directions 144
6.3.1 Domain generalisation 144
6.3.2 Multi-modal sensing and foundation models 145
6.3.3 Generative methods 146
6.3.4 Collaborative and adaptive sensing 147
6.3.5 Beyond activity recognition 148

Bibliography 149

Chapter 1

Introduction

The widespread adoption of smartphones and wearable devices over the past decade has revolutionised how people live: from distant communication and navigation to gaining knowledge about virtually any topic in an instant. In addition to active and conscious interactions with these devices, advances in sensor technologies allow sensor data to be collected efficiently in the background. The data from these devices reflect users' actions closely: instead of only being used in certain places or scenarios, these devices are carried by the user during most daily activities, and the devices experience changes in the environment at the same time as the user. This unique property opens up opportunities in the area of mobile sensing, which studies how data collected from the sensors that people carry and in the surroundings can be used to understand and model user behaviours.

Within mobile sensing, human activity recognition (HAR) is a fundamental task enabling higher-level applications. It involves identifying the actions performed by a person based on the data captured by sensors as influenced by the person and the surroundings. The contextual information extracted by the HAR system is an essential component of applications that need to be aware of the users' behaviours. For example, in healthcare, fall detection and metabolic energy expenditure monitoring can be inferred from user activities [Liang et al., 2014, Ghayvat et al., 2015, de la Concepción et al., 2017]. In industrial settings, information about user activities can be used to perform quality assurance and assist the manufacturing process [Bader et al., 2015, Xia et al., 2020]. It also has applications in sports and exercise tracking [Ladha et al., 2013], smart environments [Jalal et al., 2014] and many more.

In order to model the relationship between sensor data and human activities, a powerful feature extractor and classification algorithm are some of the most important components of HAR systems. Traditional feature-engineering approaches often require expert domain knowledge and rely on human intuition, which limits the performance and generalisability of HAR systems. Coinciding with the boom in deep learning research and deployments for various applications, which have demonstrated human-level performance, researchers also started to apply deep learning methods to mobile sensing tasks. They have been able to show superior performance compared to traditional machine learning methods with feature engineering, and show improvements by utilising increasing amounts of data [Zeng et al., 2014, Yang et al., 2015, Ronao and Cho, 2016, Lee et al., 2017, Saeed et al., 2019]. However, modelling sensor data using deep learning remains a difficult challenge due to issues inherent in mobile sensing.

1.1 Motivation

Several unique challenges arise when using deep learning methods for human activity recognition, which are less prominent in other computational tasks and applications: the constraints on computational resources, especially in the context of power management, the computing speed, the large variations between different mobile devices and sensors, the differences in physical characteristics among individual users, and most importantly the significantly higher difficulty in collecting labelled data for mobile sensing tasks [Saeed et al., 2019]. Although deep learning methods can automatically learn to extract useful features, the bias and limitations present in datasets, especially in their limited ability to reflect the various scenarios in the real world, have negative effects on the real-world performance of the models [Torralba and Efros, 2011, Tommasi et al., 2017, Mehrabi et al., 2021].

For many other tasks, such as image classification and language understanding, it is often possible to label the data after collection since most of these tasks are intuitive to humans. This enables data to be gathered from a variety of places, settings and sources before labelling. However, mobile sensing tasks including human activity recognition face additional challenges. Without the aid of external devices or tools, such as cameras or microphones, HAR often requires collecting labels at the same time as the sensor data. Research participants are typically required to follow strict protocols to perform a scripted set of actions during data collection [Stisen et al., 2015, Chatzaki et al., 2016, Malekzadeh et al., 2018] so that labels can be obtained. Attempting to label raw sensor signals post-collection by inspection is very difficult or virtually impossible, unlike more intuitive tasks. Many labelled datasets that were used to construct and validate HAR systems have significant limitations. They often come from less than 30 participants over a few hours in a laboratory environment, and hence exhibit limitations in terms of quantity, data diversity, and real-world generalisation of human activity in everyday life. In contrast, these limitations are less pronounced in datasets in other deep learning domains.

Furthermore, there is often a lack of universal agreement on the selections, models, placements and capabilities of sensors that are embedded in mobile devices. This poses a challenge in mobile sensing where there is an enormous number of variations in usage among users. For instance, some users might have multiple wearable devices, while many only use a smartphone. This leads to relatively low performance in HAR if we simply train one model on a particular device and a particular set of sensors, and transfer it to a different setting. Furthermore, the aforementioned difficulty of collecting labelled data in mobile sensing readily applies here: it is not scalable and economical to capture all possible setups of sensors. Even though smartphones often share a similar set of basic sensors such as accelerometer, gyroscope, GPS and others, differences between manufacturers and batches can mean that models need to be retrained on new datasets from a wide range of users whenever a new device is developed, in order to maintain the same level of performance, which exacerbates the scalability problem.

In addition, although using multiple sensors and devices tends to improve the performance of mobile sensing systems [Shoaib et al., 2016, Vaizman et al., 2017, Wang et al., 2019], the improvement usually comes with a cost: the increase in power consumption of processing and transmitting sensor data, and the need for more computational resources of using more complex models which incorporate multiple data sources. The limit of power resources on smartphones and wearable devices limits the performance of mobile sensing systems, which leads to a trade-off between accuracy and power consumption [Yan et al., 2012, Bulling et al., 2014].

With the advantages of mobile devices being ubiquitous and being near users most of the time, and the disadvantages of having constraints on resources and difficulty of collecting labelled datasets, semi-supervised and self-supervised learning methods offer a unique opportunity to mitigate the disadvantages while utilising the advantages. The use cases of mobile devices allow data from one or more devices to be passively collected in the background, without involving user interactions. Although these data do not come with labels, it is possible to collect a large quantity of them at a very low cost, and they are closely tied to the physical characteristics of the user and the environment that the user is in. Self-supervised and semi-supervised learning methods become important tools to overcome the limitation of labelled datasets by utilising other sources of data, including the abundant unlabelled data and data from multiple devices.

1.2 Thesis and substantiation

The main objective of my research is the exploration and development of self-supervised and semi-supervised machine learning techniques which leverage easily obtainable data, for the development of scalable and generalisable mobile sensing models which can effectively adapt to different settings in human activity recognition. The aim is to extend existing methods to perform human activity recognition with different users and setups of devices without the collection of large-scale labelled data.

In particular, the thesis aims to answer the following research questions:

Research Question 1. In the area of wearable-based human activity recognition systems, where unlabelled data is abundant while collecting data encompassing various use cases proves challenging, can unlabelled data be efficiently combined with a limited amount of labelled data, to develop scalable human activity recognition systems?

Research Question 2. By extension, in multi-device mobile sensing setups, in which users carry multiple mobile devices, can data from other devices be utilised to develop more accurate HAR models?

Research Question 3. In continual learning scenarios where the model is required to adapt to changing data distributions, can multi-task learning and self-supervised learning methods be used to train more generalisable HAR models and mitigate catastrophic forgetting?

To address these questions, we designed novel training paradigms for HAR systems based on self-supervised learning and other deep learning techniques that can leverage large-scale unlabelled sensing data, make use of data from multiple devices, and are better at handling changing data distributions.

1.3 Contributions and chapter outline

This thesis will start with an overview of the background and existing works in human activity recognition in Chapter 2, before a presentation of the three main contributions involving novel training paradigms for HAR in Chapters 3, 4 and 5:

Contribution 1: Improving human activity recognition through self-supervised learning and self-training with unlabelled data

Chapter 3 looks into how self-supervised learning, together with self-training, can be an effective way to leverage unlabelled data in the development of human activity recognition systems.

The proposed method, named *SelfHAR*, combines the advantages of self-supervised learning, which increases the diversity of data that the model is trained on through data augmentation, and those of self-training, where large-scale unlabelled datasets can be efficiently used for knowledge distillation. *SelfHAR* adopts a teacher-student knowledge distillation setup [Zhu, 2005, Van Engelen and Hoos, 2020], where a teacher model is first trained, and the model's knowledge is then distilled by labelling a large unlabelled dataset. Data samples with high confidence are selected and then augmented using signal transformation functions to increase the diversity of data. The student model is first pre-trained in a multi-task setup involving the tasks of recognising which transformation function has been applied and which activity each sample corresponds to. The student model is lastly fine-tuned with ground-truth labels from the training set.

We validated the proposed method with several publicly available datasets, as well as a large-scale unlabelled dataset from over 2000 participants, and the experiment results showed state-of-the-art performance over existing supervised and semi-supervised approaches, with up to 12% increase in F_1 score using the same number of model parameters at inference. Furthermore, *SelfHAR* was shown to be data-efficient, reaching similar levels of performance using up to 10 times less labelled data compared to supervised approaches. Our proposal and findings contribute to the development of human activity recognition systems that use less labelled data, which addresses one of the most important challenges in mobile sensing. The work also demonstrated the potential of leveraging self-training as a method for pre-training HAR systems with large-scale unlabelled data.

Contribution 2: Collaborative contrastive learning for human activity recognition

Expanding on the previous work, in Chapter 4 we look at how self-supervised learning can be used in a multi-device setting. We study the problem setting of Time-Synchronous Multi-Device Systems (TSMDS), in which multiple devices all observe the same physical phenomenon (such as a user's physical activity) and sensor data are recorded in a time-aligned manner. We investigate how contrastive self-supervised learning can be useful in utilising multi-device data in this setting.

We proposed a novel training paradigm called Collaborative Self-Supervised Learning (*ColloSSL*) which makes use of unlabelled data collected from *multiple* devices worn by a user to learn high-quality representations of the data. The insight that inspires the design of the proposed approach is that data from multiple devices carried by the same user represent different views of the same physical activity and these data can be seen as natural transformations of each other. This observation is well-fitting to the setup of contrastive self-supervised learning, in which the training task is to make representations of different views of the same data sample similar while making those from different data samples different. Instead of using manually chosen augmentation functions to generate different views, we use data from different devices to set up contrastive learning tasks. We present three novel techniques for extending self-supervised learning methods to a multi-device setting: a *Device Selection* scheme which is used to select the data sources for forming positive and negative pairs in contrastive learning, a *Contrastive Sampling* algorithm which specifies how data from these positive and negative devices are sampled, and the *Multi-view Contrastive Loss* which is an extension of one of the commonly used loss functions in contrastive learning to the multi-view setting.

We validated different aspects of our proposal using several publicly available datasets, and we found that *ColloSSL* outperformed both fully supervised and existing semi-supervised learning techniques in the majority of the experiment settings. In addition to performance, we also demonstrated that *ColloSSL* is data-efficient, outperforming simple fully-supervised learning with less than one-tenth of the labelled data in most experiments. The proposed method was shown to be robust against common sensor problems including missing data and time synchronisation errors. This work demonstrated that unlabelled data from multiple devices can be used to overcome the limitations of labelled datasets, and the potential of contrastive learning as a data-efficient training method.

Contribution 3: Overcoming catastrophic forgetting in continual learning with multi-task and self-supervised learning

In Chapter 5, we investigate continual learning settings where models are expected to perform well in the presence of data distribution shifts. We specifically look at class-incremental learning, which is a setting under continual learning, and how multi-task learning and self-supervised can be used to overcome some of the challenges. In this setting, new classes in a classification problem are introduced from time to time, and the task is that the model should perform well in all the tasks that it has been trained on. A prevalent problem in continual learning is catastrophic forgetting, in which models forget what they have learned after being re-trained on new data.

The chapter begins with an investigation into how multi-task learning during early model training can be used to improve the feature generalisability, and hence allow models to be better equipped to learn from new tasks later on. Specifically, we designed a novel training paradigm for the initial model training step, where the base classes are decomposed into subsets and the model is trained simultaneously on multiple tasks. This setup simulates the scenario of continual learning at the initial training step, in which the model is trained to balance the objective of being performant in multiple data distributions. The approach was evaluated on a publicly available dataset, and results showed that it improved the average incremental learning accuracy by up to 6.4%, enabling more reliable and accurate activity recognition over time.

In addition to improving the feature generalisability at the initial training step, the second part of the chapter looks into how continual self-supervised learning can be generalised to a more practical setting, where unlabelled and labelled data are leveraged to balance different learning objectives. By extending existing works in continual self-supervised learning for feature learning, we propose a novel training architecture which mitigates catastrophic forgetting for both the feature extractor and the classifier with a carefully designed loss function. The overall loss function consists of four main components, corresponding to knowledge distillation and current task learning, one of each for the feature extractor and the classifier. The proposed method addresses the shortcomings of existing works in continual self-supervised learning, which requires a two-step training process for the target classification task, by using a unified scheme in which a classifier is continually trained in conjunction with the feature extractor, allowing deployment at any point during the training process. The use of self-supervised learning was proven powerful for knowledge retention, and we demonstrated that the proposed method outperformed existing self-supervised learning models in competitive benchmarks. The proposal balances the trade-off between knowledge retention and learning for new tasks with an end-to-end model and demonstrates a promising direction for the practical deployment of continual learning systems.

Lastly, a reflection on the results and insights from the aforementioned works, as well as a discussion about future research directions is presented in the last chapter, Chapter 6, of this thesis.

1.4 List of publications

I have been fortunate to undertake fruitful collaborations with other researchers from the same research team as well as outside, which have resulted in several publications in mobile sensing and beyond. The publications arising from the work presented in this thesis have already been cited by many other researchers, with multiple studies building on top of our work.

Chapter 3 of this thesis is based on the work published in the Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT) 2021 [Tang et al., 2021], and Chapter 4 draws from the work published in IMWUT 2022 [Jain et al., 2022]. Chapter 5 builds on the work presented at the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) 2022 [Ma et al., 2022], and research presented at the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) 2024 [Tang et al., 2024a] and AAAI Conference on Artificial Intelligence 2024, Human-Centric Representation Learning workshop [Tang et al., 2024b]. I have also co-authored works that are closely related to these works and inspired some of my research.

Works related to this thesis

- [Tang et al., 2021] **Tang, C. I.**, Perez-Pozuelo, I., Spathis, D., Brage, S., Wareham, N., and Mascolo, C. (2021). Selfhar: Improving human activity recognition through self-training with unlabeled data. *Proceedings of the ACM on interactive, mobile,* *wearable and ubiquitous technologies*, 5(1):1–30.

- [Jain et al., 2022] Jain, Y.*, **Tang, C. I.***, Min, C., Kawsar, F., and Mathur, A. (2022). Collossl: Collaborative self-supervised learning for human activity recognition. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 6(1):1–28. *equal contribution

- [Ma et al., 2022] Ma, D.*, **Tang, C. I.***, and Mascolo, C. (2022). Improving feature generalizability with multitask learning in class incremental learning. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4173–4177. IEEE. *equal contribution

- [Tang et al., 2024a] **Tang, C. I.**, Qendro, L., Spathis, D., Kawsar, F., Mascolo, C., and Mathur, A. (2024a). Kaizen: Practical self-supervised continual learning with continual fine-tuning. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pages 2841–2850.

- [Tang et al., 2024b] **Tang, C. I.**, Qendro, L., Spathis, D., Kawsar, F., Mathur, A., and Mascolo, C. (2024b). Balancing Continual Learning and Fine-tuning for Human Activity Recognition. *AAAI Conference on Artificial Intelligence, Human-Centric Representation Learning workshop (AAAI-W), Vancouver, Canada.*

Other works

- [Tang et al., 2020] **Tang, C. I.**, Perez-Pozuelo, I., Spathis, D., and Mascolo, C. (2020). Exploring contrastive learning in human activity recognition for healthcare. *Advances in Neural Information Processing Systems, Machine Learning for Mobile Health workshop (NeurIPS-W), Virtual event, Canada.*

- [Lubana et al., 2022] Lubana, E., **Tang, C. I.**, Kawsar, F., Dick, R., and Mathur, A. (2022). Orchestra: Unsupervised federated learning via globally consistent clustering. In *International Conference on Machine Learning*, pages 14461–14484. PMLR.

- [Haresamudram et al., 2023] Haresamudram, H., **Tang, C. I.**, Suh, S., Lukowicz, P., and Ploetz, T. (2023). Solving the sensor-based activity recognition problem (soar): Self-supervised, multi-modal recognition of activities from wearable sensors. In *Adjunct Proceedings of the 2023 ACM International Joint Conference on Pervasive and Ubiquitous Computing & the 2023 ACM International Symposium on Wearable Computing*, pages 759–761.

- [Shah et al., 2021] Shah, K., Spathis, D., **Tang, C. I.**, and Mascolo, C. (2021). Evaluating contrastive learning on wearable timeseries for downstream clinical outcomes. *Machine Learning for Health (ML4H).*

- [Luo et al., 2020] Luo, R., Wong, C.-L., Wong, Y.-S., **Tang, C. I.**, Liu, C.-M., Leung, C.-M., and Lam, T.-W. (2020). Exploring the limit of using a deep neural network on pileup data for germline variant calling. *Nature Machine Intelligence*, 2(4):220–227.

Chapter 2

Background

2.1 Human activity recognition

Human activity recognition (HAR) involves identifying actions performed by a person based on sensor data capturing the person's movements and surroundings. The activities to be recognised typically reflect common human motions, while the sensor data can come from various sources.

The complexity of activities for recognition can vary depending on the use case. It can range from simple actions such as raising one's hand and nodding, to complex multi-step activities like cooking and shopping. [Moeslund et al., 2006] proposed a hierarchy of activity abstractions based on the level of human involvement: *action primitives* describe simple limb movements (e.g. right leg forward), *actions* describe whole-body movements (e.g. walking), and *activities* interpret sequences of movements (e.g. jumping hurdles). Naturally, activities of different complexity involve different time-frames, requiring tailored algorithm designs. This thesis focuses on recognising activities of daily living (ADL) [Reiss and Stricker, 2012, Anguita et al., 2013, Micucci et al., 2017], corresponding to *actions* or *activities* as mentioned above.

HAR systems can also be broadly categorised into two main types: sensor-based HAR and video-based HAR [Ranasinghe et al., 2016]. Video-based approaches benefit from the widespread deployment of video surveillance systems, providing a holistic view of the surroundings, a steady stream of data, and computational resources [Ranasinghe et al., 2016, Sabokrou et al., 2018]. This makes them well-suited for security and surveillance applications. However, major privacy, pervasiveness, and complexity concerns exist in these methods [Lara and Labrador, 2013, Ranasinghe et al., 2016], limiting scalability and usability, especially for personal use cases. In contrast, sensor-based approaches directly leverage signals from on-body sensors, avoiding such issues.

2.1.1 Sensor-based HAR

Sensor-based HAR systems using embedded sensors in mobile devices have several beneficial properties that make them suitable for personal mobile computing: high pervasiveness, privacy guarantees, and low complexity. Performing computation locally on the device enables non-intrusive data collection and privacy. Furthermore, the pervasiveness of these systems, which enables the collection of real-time user behaviour data continuously, is particularly useful in healthcare applications [Menschner et al., 2011], allowing for individualised monitoring. Sensor data also tend to have lower complexity and variance compared to other data modalities such as videos, allowing for simpler data processing [Lara and Labrador, 2013].

The typical sensor-based HAR workflow, called the Activity Recognition Chain [Bulling et al., 2014], and by extension, many other mobile sensing systems, involve five main steps: data acquisition, signal preprocessing, segmentation, feature extraction and selection, and training and inference. Depending on the application and use cases, many design choices can be made at different stages of this chain: selecting which sensor and which device to acquire data from, which signal preprocessing methods to use, how data are segmented into windows, selecting the appropriate feature extraction and modelling method, and finally how to train and use models for classification.

2.1.2 Data acquisition

Inertial measurement units (IMUs) embedded in mobile devices are the most common data source for sensor-based HAR due to their widespread adoption and relatively low power requirements. Using accelerometer data from a single device is one of the simplest approaches, which has been studied extensively in existing research [Casale et al., 2011, Bayat et al., 2014, Ignatov, 2018, Nweke et al., 2018, Bento et al., 2023]. Other IMUs such as the gyroscope and magnetometers are also frequently used alongside accelerometers because they are often implemented in the same module [Altun and Barshan, 2010, Ferrari et al., 2019, Masum et al., 2019, Webber and Rojas, 2021]. Additional sensors such as GPS, Bluetooth, Wi-Fi, microphones, proximity and light sensors can also be used for recognising more complex activities within different contexts [Wang et al., 2009, Martín et al., 2013, Nweke et al., 2018, Wang et al., 2019]. However, there is often a trade-off between accuracy and power efficiency, because more active sensors provide more contextual clues but also consume more power.

Collaborative and adaptive sensing

Beyond single-device sensing, using data from multiple devices is also possible. Different devices capture user activities from different viewpoints, and each of them comes with different interaction patterns and capabilities. For example, smartphones often have the most computing resources including processing power and battery life, while smartwatches provide less noisy data due to their fixed wearing position. These complementary strengths and weaknesses of devices, along with their multiple perspectives on user behaviour, have motivated research into collaborative and adaptive activity recognition strategies, aiming to achieve better accuracy and efficiency [Zappi et al., 2008, Kang et al., 2010, Keally et al., 2011, Shoaib et al., 2016, Vaizman et al., 2017, Min et al., 2019].

One popular topic in this area is sensor selection strategies. They have been proposed to maximise the system utility in body sensor networks, by dynamically selecting the best sensors based on device parameters, such as accuracy, resource usage, and device availability [Zappi et al., 2008, Wang et al., 2009, Kang et al., 2010, Keally et al., 2011, Min et al., 2019]. The work by [Bao and Intille, 2004] pointed out that data from thigh and wrist accelerometers have the strongest predicting ability for recognising daily activities. A dynamic sensor selection scheme which turns on the minimal set of sensors to satisfy accuracy requirements has been proposed by [Zappi et al., 2008], while [Wang et al., 2009] introduced a flexible sensor management framework allowing system designers to define state transition between activities. [Yan et al., 2012] looked at selecting different preprocessing functions and sampling frequencies based on the detected activity, and [Min et al., 2019] proposed a duty-cycle system based on runtime quality assessment functions. While these strategies focus on providing better runtime system performance in a multi-device environment, they do not leverage data across devices to improve the accuracy of activity recognition.

To improve accuracy, sensor fusion techniques have been proposed to leverage data from multiple devices [Ordóñez and Roggen, 2016, Shoaib et al., 2016, Vaizman et al., 2017, Yao et al., 2017, Peng et al., 2018, Vaizman et al., 2018, Yao et al., 2018a]. Studies such as those by [Shoaib et al., 2016, Vaizman et al., 2018] have demonstrated higher effectiveness of using data from devices placed at the thigh and the wrist (corresponding to smartphones and smartwatches, which is one of the most common settings) compared to using a single device. In these works, multiple sensor streams are concatenated as features for better accuracy performance.

These collaborative and adaptive sensing strategies explore the aforementioned trade-off between accuracy and power efficiency, shining light on how different use cases can be realised and how user experience can be improved.

2.1.3 Modelling sensor time-series

After acquiring and preprocessing sensor data, feature extraction for modelling sensor data is one of the most important components in the activity recognition chain. Over the past decade, feature extraction methods have seen significant shifts from traditional feature engineering and simple machine learning methods to neural networks, coinciding with advances in deep learning research.

Traditional feature-engineering approaches

Time domain and frequency domain metrics are traditionally used as features for HAR [Huynh and Schiele, 2005, Figo et al., 2010, Plötz et al., 2011, Chen and Shen, 2017]. After sensor data are segmented into windows using a sliding window procedure, time domain statistical features such as mean, standard deviation, energy, entropy, and correlation coefficients, as well as frequency domain features such as Fourier coefficients and wavelet coefficients are extracted as features. Representation in the form of empirical cumulative distribution functions (ECDFs) can also be used as an alternative, which can allow better preservation of statistical features across different data ranges [Plötz et al., 2011, Hammerla et al., 2013]. Different features were shown to be effective for recognising different activities and selecting the right features is dependent on the use cases [Huynh and Schiele, 2005].

For the final task of activity recognition, which is a classification task, a classifier based on traditional machine learning methods is often developed. Algorithms including k-nearest neighbour, random forests, support vector machines (SVMs), and Naïve Bayes are commonly used [Guan et al., 2007, Chen and Shen, 2017]. Workbenches such as WEKA [Hall et al., 2009] are often useful tools for exploring different classification algorithms.

While demonstrating moderate success in modelling time series and activity recognition, these methods are limited by the inability to leverage larger amounts of data, as well as adapt to different use-case scenarios.

Artificial neural networks

Artificial neural networks are mathematical models consisting of a collection of artificial neurons in a connected manner. Artificial neurons are often implemented as *Perceptrons*, an early model of biological neurons and information organisation in the brain [McCulloch and Pitts, 1943, Rosenblatt, 1958]. In machine learning, these networks model the relationship between data, which are often categorised into inputs and outputs, and these networks act as function approximators.

A single neuron in the form of a perceptron implements a simple linear projection, followed by an activation function:

$$f(\boldsymbol{x}) = g(\boldsymbol{w}^\top \boldsymbol{x} + b)$$

where $\boldsymbol{x}$ represents the input vector, $f(\boldsymbol{x})$ represents the output of the neuron, $\boldsymbol{w}$ represents the learnable weights of the neuron, b is the bias term, and g is the activation function. The activation function introduces non-linearity into a neural network since multiple linear projections can be collapsed into a single linear projection.

There are two broad types of neural networks: feed-forward and recurrent networks. If we view neural networks as directed graphs, with nodes representing neurons, and edges representing connections in which the activation of an outgoing neuron is passed to the incoming neuron, feed-forward networks are directed acyclic graphs while recurrent networks contain self-loops.

Neurons are often organised into layers with no connections between neurons within the same layer, and one layer's output connects to another layer's input. A simple neural network with two hidden layers can be implemented as follows:

$$\begin{aligned} f(\boldsymbol{x}) &= f^{(3)}(f^{(2)}(f^{(1)}(\boldsymbol{x}))) \\ f^{(1)}(\boldsymbol{x}) &= g_1(\boldsymbol{W_1 x} + \boldsymbol{b_1}) \\ f^{(2)}(\boldsymbol{x}) &= g_2(\boldsymbol{W_2 x} + \boldsymbol{b_2}) \\ f^{(3)}(\boldsymbol{x}) &= g_3(\boldsymbol{W_3 x} + \boldsymbol{b_3}) \end{aligned} \tag{2.1}$$

where $f^{(i)}(\boldsymbol{x})$ represent the output of the entire layer i, $\boldsymbol{W_i}$ represent the weights as a matrix, $\boldsymbol{b_i}$ are the bias vectors, and g_i are the activation functions. The final activation function g_3 is often different from the other ones because the outputs of the final layer are used to model the existing data (which are called labels, ground truths, and annotations). For classification tasks, a *softmax* function is often used to produce a probability distribution, while for regression problems, it is common to use the identity function.

It can be seen that increasing the number of layers increases the complexity of the model, and *deep learning* refers to learning algorithms using neural networks with many layers and a deep architecture.

The performance of artificial neural networks is quantified by a loss function. This function measures how well the outputs from the neural network match the labels (that is, the annotation in the dataset). A simple example of a loss function is the mean squared error:

$$\mathcal{L}_{\text{MSE}}(Y, \hat{Y}) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{2.2}$$

where $\hat{Y}$ is the set of predictions from a neural network, Y is the ground truth, Y_i and $\hat{Y}_i$ are the i-th element of these sets, and n is the number of training samples. For classification problems, which are more relevant to HAR systems, the multi-class cross-entropy loss function is commonly used:

$$\mathcal{L}_{\text{CE}}(Y, \hat{Y}) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{C} Y_{ik}\ \log(\hat{Y}_{ik}) \tag{2.3}$$

where C is the number of classes, $\hat{Y}_{ik}$ is the prediction probability for a particular class

k on the i-th sample, and Y_{ik} is the ground-truth probability for that class of the same sample (usually set to the value 1 for the true class, 0 for the rest).

Training an artificial neural network involves minimising the loss. The conventional training algorithm is Gradient Descent [Lemaréchal, 2012]:

$$W := W - \gamma \frac{\partial \mathcal{L}}{\partial W} \tag{2.4}$$

where W represents the learnable parameters of the network, γ is the learning rate, and $\mathcal{L}$ is the loss function. Since all the components of a neural network are differentiable, the chain rule from calculus can be applied, and this results in the back-propagation algorithm [Rosenblatt et al., 1962, Rumelhart et al., 1986], where the forward pass involves passing the inputs and activations through the neural network layer by layer, and the backward pass applies the chain rule and calculates the gradient updates in the reverse order needed to minimise the loss. [Goodfellow et al., 2016] wrote a good source of reference for a more in-depth discussion about deep learning.

Overall, neural networks are powerful function approximators whose expressiveness comes from combining many simple neurons into deep architectures. Deep networks with many layers can represent sophisticated relationships between inputs and outputs.

Architectures of neural networks

Although the fully-connected architecture based on perceptrons was shown to be powerful [Gardner and Dorling, 1998], they are also prone to overfitting, in which the model only learns to perform well on training samples, but fails to generalise to unseen inputs [Li et al., 2019]. Convolutional neural networks (also called Shift-Invariant Artificial Neural Networks [Zhang et al., 1988] and Time-Delay Neural Networks [Waibel, 1989]) offer an alternative architecture for neural networks, especially for modelling images and sequence data, in which parameters are re-used in the form of convolutional kernels or filters instead of being 'fully-connected' between layers in the perceptron setup [LeCun et al., 1989]. These learnable kernels and filters are applied repeatedly across different dimensions of the input and mimic the convolution or correlation operations in image processing [Jacobs, 2005]:

$$F \circ I(x) = \sum_{i=-N}^{N} F(i)\ I(x+i) \tag{2.5}$$

where F refers to the correlation filter, I refers to the input, and $2N+1$ is the size of the filter. Stacking several convolutional layers creates a hierarchy of features of increasing abstractness since filters in deeper layers work with higher-level features extracted by previous layers [Ordóñez and Roggen, 2016]. This architecture achieved significant success in improving over previous object recognition efforts [Krizhevsky et al., 2012] and attracted immense interest to the machine learning community.

Another variant of neural network architectures is recurrent neural networks [Medsker and Jain, 1999], which contain cycles in the computation graph. They are designed to model sequence data, in which data samples are typically fed one by one into the network, instead of being flattened and fed to a simple fully-connected layer. The classical setup for these networks involves the definition of hidden states, and they are updated as the network processes different timestamps of data [Zaremba et al., 2014]. The transition function for the hidden states from $\boldsymbol{h_{t-1}}$ to $\boldsymbol{h_t}$ is typically in the following form:

$$\boldsymbol{h_t} = f((\boldsymbol{W_h}\ \boldsymbol{h_{t-1}} + \boldsymbol{b_h}) + (\boldsymbol{W_x}\ \boldsymbol{x_t} + \boldsymbol{b_x})) \tag{2.6}$$

where $\boldsymbol{h_t}$ represents the hidden states at timestamp t, $\boldsymbol{W_h}, \boldsymbol{b_h}, \boldsymbol{W_x}, \boldsymbol{b_x}$ are learnable parameters, and $\boldsymbol{x_t}$ is the input data (or activations from the previous layer). However, works [Hochreiter and Schmidhuber, 1997, Graves et al., 2008, Cho et al., 2014, Zaremba et al., 2014] have shown that stronger regularisation methods are needed in order to successfully train a recurrent neural network. Recurrent layers with Long Short-Term Memory (LSTM) units [Hochreiter and Schmidhuber, 1997] have been proposed to better structure the inner workings of the recurrent units, with a design to mimic memory cells. They have been shown to be very successful in modelling sequence data including audio [Sak et al., 2014] and text [Stahlberg, 2020].

In order to model sensor time series, it is natural to make use of recurrent architectures since they are designed to model sequence data. Works by [Chen et al., 2016, Pienaar and Malekian, 2019] have seen success in adopting such architectures for the task of human activity recognition, in which the time dependency within sensor data is modelled using recurrent layers. On the other hand, although the convolutional architecture is often applied in the area of image processing, in which inputs are two-dimensional, works have shown that they can also be successfully applied to one-dimensional sequence data [van den Oord et al., 2016, Dauphin et al., 2017]. This includes human activity recognition, where works [Zeng et al., 2014, Yang et al., 2015, Ronao and Cho, 2016, Lee et al., 2017, Peng et al., 2018, Saeed et al., 2019, Zhai et al., 2020] have demonstrated that convolutional neural networks can offer better performance and higher efficiency in modelling sensor time series, compared to recurrent architectures. [Yang et al., 2015] demonstrated strong performance by using neural networks with temporal convolutions (1D convolution) for automatically extracting useful features from raw sensor signals for human activity recognition tasks. The ability to model sensor data over different time scales and their shift-invariant nature allow them to model sensor data successfully.

DeepConvLSTM [Ordóñez and Roggen, 2016] is an architecture designed for human activity recognition that combines convolutional and recurrent layers. This architecture aims to combine the strengths of two different types of layers: convolutional layers are shift-invariant and extract features of different levels of abstractness, while recurrent layers capture the time dependencies in sensor data. The proposed architecture is composed of four consecutive convolutional layers, followed by two recurrent layers and another fully-connected layer for classification. The authors demonstrated state-of-the-art results at the time of publication and showed that such architecture is capable of modelling periodic and sporadic activities and that raw sensor signals with minimal pre-processing can be modelled by the network.

Recent works [Bai et al., 2018, Haresamudram et al., 2022] have compared the performance between different architectures and re-examined the effectiveness of convolutional neural networks. [Bai et al., 2018] proposed temporal convolutional networks, which are composed of pure convolutional layers in increasing perceptive fields with increasing dilation factors as the model gets deeper. As part of the baseline evaluation, [Haresamudram et al., 2022] compared different classifier architectures including LSTM, DeepConvLSTM and pure convolutional networks. They have demonstrated that while recurrent layers can model time dependencies well, convolutional architectures can often match the performance while being easier to train and efficient to run. Since efficiency is a key design consideration in mobile systems, convolutional networks are well-suited for sensor-based HAR. Therefore, this thesis will mainly leverage convolutional architectures.

2.2 Training paradigms

The training procedures introduced in the previous section describe how a neural network can learn to classify certain activities through supervised learning: the model is purely trained on samples with human annotations. Although deep learning has proven effective in extracting useful features from laboratory datasets, the performance of both traditional HAR methods and deep learning models in real-world settings could suffer due to the biases and limitations introduced by conventional laboratory-based HAR datasets. Specifically, real-world performance depends on the size, diversity, and representativeness of training data with respect to the true characteristics of the activities [Torralba and Efros, 2011, Tommasi et al., 2017, Mehrabi et al., 2021], which are often lacking from conventional laboratory-based HAR datasets. This problem is amplified in HAR due to the inherent difficulty in the collection of labelled sensor data, as discussed in Chapter 1.

Apart from dedicating more effort to collecting high-quality datasets, we can explore ways to leverage labelled data more efficiently. In particular, given the relative ease of collection and abundance of unlabelled data, this thesis aims to explore alternative training paradigms including semi-supervised learning, self-supervised learning, contrastive learning, multi-task learning, and self-training, for developing data-efficient HAR systems.

2.2.1 Semi-supervised learning

Semi-supervised learning, as the name suggests, refers to the training paradigm in which models learn from both labelled and unlabelled samples. It includes a broad range of methods that aims to utilise unlabelled data to complement and circumvent the limitations of using only labelled data by improving the quantity and diversity of training data [Zhu, 2005, Stikic et al., 2008, Zhu and Goldberg, 2009, Van Engelen and Hoos, 2020]. Despite being unlabelled, these free-living, unconstrained data have advantages in terms of their size, diversity and ability to capture and represent the richness, noise and complexity of sensor data in the real world [Torralba and Efros, 2011, Tommasi et al., 2017, Mehrabi et al., 2021]. Autoencoders [Vincent et al., 2008] and generative methods [Kingma et al., 2014, Yao et al., 2018b] are some of the methods aiming to mitigate the inherent limitations associated with supervised learning techniques.

In human activity recognition, due to the difficulty in collecting labelled sensor data, semi-supervised learning has been actively researched [Guan et al., 2007, Bhattacharya et al., 2014, Saeed et al., 2019]. The *En-Co-training* [Guan et al., 2007] algorithm combines co-training and ensemble learning to leverage unlabelled sensor data. The algorithm iteratively trains an ensemble of several different classifiers (an ensemble of a decision tree, a Naïve-Bayes classifier and a k-nearest neighbour classifier was suggested by the authors), and uses the trained classifiers to generate labels (i.e. pseudo-labels) for a pool of unlabelled samples. The samples on which all classifiers agree are put into the training set. After a predefined number of iterations, the final predictions are obtained by majority voting from the ensemble. Although the paradigm can incorporate unlabelled data, the iterative approach requires training many models from scratch, and this becomes costly when the amount of unlabelled and labelled data is large. Also, the diversity requirement limits the choice of classifiers, where they need to be sufficiently different from each other to provide a proper supervisory signal.

In another work, [Bhattacharya et al., 2014] proposed a sparse-coding framework for using unlabelled data in HAR. It derives a dictionary of basis vectors from unlabelled data to linearly reconstruct signals. Filtered by empirical entropy, these basis vectors decompose new samples into linear coefficients (activations) that are used as features. A classifier, such as a support vector machine, is then trained on these features for HAR. State-of-the-art results were reported in the study, but the requirement of solving regularised least square problems during dictionary learning and prediction makes it difficult to scale to a large amount of data.

2.2.2 Self-supervised learning

Similar to semi-supervised training, self-supervised learning is a training paradigm in which unlabelled data can be used to train a network. Artificial tasks (also called pretext tasks) are specifically designed to *generate* labels for unlabelled data for training. It has been actively studied due to its simplicity, similarity to supervised methods, and the ability to exploit the invariants and properties of the data itself to provide a supervisory signal.

Self-supervised learning has been extensively studied in the wider machine learning community. [Jing and Tian, 2020] categorised self-supervised learning methods into four categories: (1) Generation-based methods, in which models are trained to generate data, (2) Context-based methods, where tasks are designed such that they capture the context structure or similarity, (3) Free semantic label-based methods, where the task is to imitate and learn to predict labels generated by traditional algorithms and (4) Cross modal-based methods, where models are trained to verify the correspondence of data of two different modalities. This categorisation also readily applies to mobile sensing.

Generative methods

Autoencoders, where the model is trained to reconstruct the input after going through a bottleneck layer, is one of the classic examples of generative methods [Vincent et al., 2008]. The input, which is often high-dimensional, is encoded to a lower-dimensional representation using an encoder and then decoded back to recreate the original input through the decoder. The task is to minimise the difference between the input and the reconstructed output, which in turn should encourage the encoder to capture the most distinctive and important features in the data. This has been applied to human activity recognition in several works [Almaslukh et al., 2017, Varamin et al., 2018, Thakur et al., 2022], demonstrating effectiveness in leveraging unlabelled data.

Generative Adversarial Networks (GANs) [Goodfellow et al., 2014] are commonly used in generation-based self-supervised learning methods, due to their inherent generative nature. Variants of GANs, such as DCGAN [Radford et al., 2016] (where a deep convolutional network is trained in an adversarial setting for image generation) and SRGAN [Ledig et al., 2017] (in which a model is trained to generate high-resolution photos) have emerged. BiGAN [Donahue et al., 2017], in which an encoder is jointly trained with a generator, such that the model is able to learn to encode images into a latent feature space while also generating images from the same space, directly encodes the task of representation learning into its architecture. It is worth exploring if such methods could be effective for mobile sensing. SensoryGAN [Wang et al., 2018] is one of the few attempts at using GANs in the context of mobile sensing. However, the study mainly focused on the generation of synthetic data, without exploring the potential of using GANs for representation learning.

Occlusion, where part of the input data is hidden from the model, is also another commonly used generative technique in self-supervised learning. Inpainting, in which a model is asked to fill in hidden pixels of an image, was proposed by [Pathak et al., 2016] and is a classic example of the occlusion task. This task forces the model to learn the distribution of natural images in order to regenerate the missing part. A similar idea has also been applied to language modelling. The Bidirectional Encoder Representations from Transformers (BERT) [Devlin et al., 2019], which was shown to achieve state-of-the-art performance, includes pre-training the model to predict missing words in a sentence. In human activity recognition, [Haresamudram et al., 2020] have implemented masked reconstruction, in which the model is trained to reconstruct data points masked randomly in the input. It has demonstrated satisfactory performance when compared to unsupervised and supervised training methods.

Context-based methods

Context-based methods exploit the context structure or similarity that is inherently present in the data itself. Jigsaw puzzle-solving [Doersch et al., 2015, Noroozi and Favaro, 2016, Wei et al., 2019] is a common strategy for exploiting the spatial structure of images. Their video counterparts [Misra et al., 2016, Ahsan et al., 2019] exploit the temporal context in addition to the spatial context of videos. These temporal methods should be readily applicable to sensor data since they also exhibit temporal structures. However, a single task might not be enough to provide the supervisory signal to train models to learn the general notion of useful features.

In human activity recognition, the transformation discrimination task, in which the model is trained to recognise which transformation function has been applied to a data sample, has been proposed by [Saeed et al., 2019]. This approach was adapted to electrocardiogram (ECG) data for emotion recognition by [Sarkar and Etemad, 2020], and the authors performed an investigation into the impact on the recognition accuracy caused by changing the difficulty of the transformation tasks and using different sets of unlabelled data, as well as the relationship between downstream (target) and upstream (pre-training) tasks. They reported state-of-the-art results on different datasets compared to previously proposed approaches, using pre-training on transformation discrimination before fine-tuning on the target dataset. This work gives insights into the relationship between the difficulty of pre-training tasks and the accuracy of the models, but whether these results can be applied to HAR is yet to be studied.

Free semantic label-based methods

Similar to context-based methods, free semantic label-based methods rely on automatically generated labels from expert-designed algorithms. For example, [Li et al., 2016] uses a hard-coded edge detection algorithm and trains the model to produce the same results. [Zhang et al., 2022] proposed a training scheme for modelling time-series data by enforcing time-frequency consistency. The authors proposed to generate an alternative frequency-domain representation of the raw sensor data originally in the time domain, by using transform operators such as Fourier Transform. The task is to train the encoder for each of these two representations to generate similar representations. These works demonstrated that hard-coded algorithms can be used as pretext tasks to train powerful feature extractors. More research into algorithms tailored to capture sensor data properties for HAR is needed.

Cross modal-based methods

Given that it is common to have different types of sensors on a single mobile device, researchers have looked at how they can leverage different modalities to set up self-supervised learning tasks.

As a generalisation of the work by [Saeed et al., 2019], [Saeed et al., 2021] proposed the *Sense and Learn* framework which leverages self-supervision for representation learning on multi-modal sensor data. The authors proposed a set of eight self-supervised tasks, from which practitioners can choose according to their knowledge of the modality of data and the target tasks, or the empirical results. The set of self-supervised tasks included ones that leverage multi-modal sensor data, such as Blend Detection, which detects whether the sensor signals from different sources have been blended together, and ones which detect changes in the data, such as transformation discrimination and Odd Segment Recognition. Although a high performance was reported when the number of labelled samples was particularly limited, the fully-supervised models often outperformed the best-performing self-supervised methods in this work. Furthermore, some of the self-supervised tasks rely on multi-modal data, which might be expensive to collect in terms of system resources, especially in real time.

The work by [Spathis et al., 2021] utilised the underlying physiological relationship between heart rate responses and movements as a supervisory signal for the extraction of user-level physiological embeddings. The authors pre-trained a network to predict heart rate signals based on accelerometer data and contextual metadata such as the hour of the day. Apart from being able to use the trained model as a heart rate estimator, the representations extracted by the model can be aggregated for each user to form user-level embeddings, which were shown to outperform other methods in predicting fitness and demographic variables, such as height and BMI. This study demonstrated the ability of self-supervised models to extract useful representations for a wide range of tasks by leveraging multi-modal data. However, it does not explicitly leverage both unlabelled and labelled data, and it is not clear if this approach would improve HAR.

2.2.3 Contrastive learning

Contrastive learning is another self-supervised approach that exploits context similarity. It is based on the assumption that groups of data exhibit high intra-group similarity and low inter-group similarity, and the goal is to group similar samples together and push dissimilar samples apart in the embedding space [Bachman et al., 2019, Chen et al., 2020a].

With unlabelled data, a data sample (called the *anchor sample*) from the training dataset is taken, and a predefined perturbation (e.g., applying a random rotation) is applied to generate a transformed version of the sample (called the *positive sample*). During the training process, the anchor sample, the positive sample and other randomly selected data samples from the dataset (called *negative samples*) are fed to a neural network to obtain feature embeddings. The model training brings the anchor and positive embeddings closer while pushing the anchor and negatives apart. This forces the model to extract quality features from raw unlabelled data, which can then be used for downstream tasks.

The work by [Chen et al., 2020a] introduced SimCLR, a simple contrastive learning scheme which follows closely the training procedure described above. Different views of the same data are generated by applying different transformations, and the model is trained to maximise the agreement between views from the same data and minimise the agreement between views from different data. Despite its simplicity, this method outperforms other self-supervised learning techniques on the ImageNet dataset, a commonly used computer vision benchmark. [Tang et al., 2020] extended this framework to human activity recognition by using a set of transformation functions that are tailored to sensor time series.

One of the most important factors that underpin the performance of contrastive learning is the choice of the perturbations for which we want the model to remain invariant while being discriminative to other negative samples. Prior research [Chen et al., 2020a, Tang et al., 2020] has shown that the choice of perturbations can have a profound impact on the performance of contrastive learning. [Chen et al., 2020a] found that the top-1 accuracy of a model pre-trained with contrastive learning on the ImageNet dataset can drop dramatically from 56.3% (using an image crop and colouring transformation) to just 2.6% (only using image rotation transformation). Similarly, in human activity recognition, the F_1 score on the MotionSense dataset decreases from 87.2% when using channel shuffling followed by segment permutation, to 59.4% using negation followed by channel shuffling [Tang et al., 2020].

Contrastive Predictive Coding (CPC) [Oord et al., 2018, Haresamudram et al., 2021] is another framework which translates a predictive, generative task to a classification task, inspired by theories of the human brain in neuroscience. In this framework, an encoder is used to compress high-dimensional data into a latent space, and these compressed data are used to predict future samples. It was also shown to be an effective self-supervised training technique which achieved state-of-the-art performance when introduced.

COCOA (Cross mOdality COntrastive leArning) [Deldari et al., 2022] proposed a contrastive task among different data modalities from different sensors, devices and channels. They propose that time-aligned samples from different sensors with different data modalities can be used for the contrastive learning setup, against samples that are not aligned. The design of the training algorithm allows efficient scaling to the number of modalities, and it is shown to outperform other self-supervised and fully-supervised proposals.

Along a similar line of research, Siamese representation learning, in which learning is done through comparing anchor and positive samples only without the use of negatives, has also proven effective in providing strong supervisory signals using large unlabelled datasets in recent works [Caron et al., 2020, Chen et al., 2020a, Chen et al., 2020b, Grill et al., 2020, Caron et al., 2021, Chen and He, 2021, Zbontar et al., 2021, Bardes et al., 2022]. These methods employ mechanisms such as gradient stopping [Grill et al., 2020, Chen and He, 2021] or a clustering task [Caron et al., 2021] to replace the use of negative samples to avoid mode collapse, in which models produce degenerate solutions. They have been shown to require fewer resources during training [Grill et al., 2020], and demonstrated strong evidence in capturing semantic meaning in data [Caron et al., 2021].

2.2.4 Self-training

Teacher-student knowledge distillation is another semi-supervised training technique that can leverage large-scale unlabelled data using an iterative training process. The basic set-up of a teacher-student training pipeline consists of (1) training a teacher model with the labelled data, (2) using the trained teacher model to generate labels for the unlabelled data (i.e. pseudo-labels), and (3) training a student model on both the original labelled data and the teacher-labelled data [Zhu, 2005, Hinton et al., 2015, Van Engelen and Hoos, 2020]. In scenarios where the teacher model and the student model share the same or very similar neural network architectures, it is called *self-training*. These approaches have been applied in different applications, for instance, in word sense disambiguation [Yarowsky, 1995], in object detection [Rosenberg et al., 2005], as well as in image classification [Yalniz et al., 2019] and semantic segmentation [Zou et al., 2018].

The work by [Yalniz et al., 2019] proposed a method to leverage billions of unlabelled images for image classification by self-training. Their proposed method closely follows the basic self-training strategy with a few additional steps: (1) train a teacher model on the target labelled dataset; (2) generate self-training labels for the unlabelled images using the teacher model; (3) for each class, select the most confident predictions by the teacher model and combine them to form a self-training dataset; (4) pre-train a student model with this dataset and finally (5) fine-tune the student model with the target labelled dataset. A key component that allows this method to leverage billions of images is the selection and filtering step, which selects only the most confident predictions for self-training. This setup was shown to be effective in improving the accuracy of a range of deep learning models in different tasks, including image classification and video classification, achieving state-of-the-art performance. This study showcased an effective way to utilise large-scale unlabelled datasets. However, this work is reliant on millions of labelled images available in the ImageNet dataset and very deep neural network architectures, which are not practical in mobile sensing.

2.2.5 Transfer learning

Transfer learning is another commonly used method to mitigate the need for collecting a large labelled dataset for specific tasks. Transfer learning methods are based on the assumption that models could learn to solve new problems faster and better by leveraging previously acquired knowledge [Pan and Yang, 2009, Tan et al., 2018]. The work by [Tan et al., 2018] classified transfer learning techniques in deep learning into four categories: (1) instances-based, which involves selecting samples that are similar to the target dataset, (2) mapping-based, which aims to map data from both the source and target datasets to a common feature space, (3) network-based, which re-uses parts of another model trained with a different dataset, and (4) adversarial-based, which involves adversarial training to extract domain-independent features.

Network-based transfer learning is a commonly used method which involves models being pre-trained on a large dataset and then fine-tuned to a different task or domain [Oquab et al., 2014, Yosinski et al., 2014]. [Oquab et al., 2014] proposed a method to improve the performance of object and action classification tasks with limited data by transferring a convolutional neural network pre-trained on the ImageNet dataset. The transfer pipeline involves freezing the pre-trained layers of the model and updating only the rest of the model during the fine-tuning phase using the target dataset. This is one of the most commonly used techniques in network-based transfer learning which aims to overcome *catastrophic forgetting* [Goodfellow et al., 2013, Sun et al., 2019], where neural networks tend to 'forget' previously learned knowledge during re-training. [Oquab et al., 2014] showed that models trained with their proposed pipeline achieved state-of-the-art performance on benchmark datasets with limited data. This setup is commonly used in semi-supervised and self-supervised training, and it has been proven an effective way to balance the importance of knowledge learned from different datasets during different training phases for a deep learning model.

2.2.6 Multi-task learning

In addition to training networks with a single task, multi-task learning requires models to solve more than one task at the same time. An example in computer vision is to localise the object and perform object recognition at the same time [Girshick, 2015]. Research [Pironkov et al., 2016, Ruder, 2017, Zhang and Yang, 2018] has shown that it allows models to be more resilient against overfitting to the training dataset by producing more generalised data representations since the models are required to solve multiple objectives at once.

In human activity recognition, the aforementioned transformation discrimination framework proposed by [Saeed et al., 2019] also utilises multi-task learning, alongside self-supervised learning. In the study, a set of eight signal transformation functions: adding random noise, scaling by a random scalar, applying a random 3D rotation, inverting the signals, reversing the direction of time, randomly scrambling sections of the signal, stretching and warping the time-series, and shuffling the different channels, as proposed by [Um et al., 2017], was used as the source of the supervisory signal in a multi-task setup. The model is trained to recognise if any of the transformation functions have been applied to the sample. Compared to the purely supervised training approach, they reported a performance gain by pre-training the model with the transformation discrimination task. Although they briefly explored the possibility of using an unlabelled dataset collected in different studies for pre-training, their architecture yielded little to no additional improvement when compared to using the same downstream dataset for pre-training. These results suggested that there is potential room for improvement in truly leveraging the information embedded in unlabelled data from external sources.

2.2.7 Continual learning

The training paradigms discussed so far assume a fixed target task, without adapting to significant data shifts over time. While models perform well on tasks they have been trained on, they usually lack the flexibility to adapt to new sequentially incoming data with the same or new classes [Diethe et al., 2019]. For example, in human activity recognition, an initial model can be built based on a pre-defined activity set, such as recognising running, walking, sitting and cycling. When the user wants the model to recognise new activities over time, such as rock climbing, the data-hungry nature of deep learning incurs several challenges. A research area, continual learning (CL, also called incremental learning and lifelong learning), where models are required to retain the acquired knowledge while learning new concepts, has been initiated to address these challenges [Van de Ven and Tolias, 2019].

Continual learning requires models to learn continually from an ever-changing stream of data. The main difference to conventional deep learning is the data assumption: rather than having all training data available at once, data with different distributions (classes or domains) arrive over time. In order to model data shifts, a continual learning dataset is usually made up of several tasks: each task corresponds to a different data distribution. The shifts in data distribution can be present in terms of the data domain and the label space [Hadsell et al., 2020]. Mathematically, we assume data $\mathbb{L} = (\mathbb{X}^{(t)}, \mathbb{Y}^{(t)})$, where $\mathbb{X}^{(t)}$ represents input samples for task t, and $\mathbb{Y}^{(t)}$ represents the labels for the corresponding samples for task t, which comes from distributions $P(\mathbb{X}^{(t)})$ and $P(\mathbb{Y}^{(t)})$, is available for the model to learn from, with the goal to optimise for all seen tasks while having little to no access to data $(\mathbb{X}^{(t')}, \mathbb{Y}^{(t')})$ from previous tasks $t' < t$ [De Lange et al., 2021].

Class incremental learning (CIL), which is a research area under continual learning, specifically looks into cases where new classes arrive at every continual learning step [Hsu et al., 2018, Van de Ven and Tolias, 2019, De Lange et al., 2021]. Typically, CIL can be split into two main training stages: the initial base model training, and the subsequent incremental learning with new classes. Each task contains an exclusive subset of classes from a dataset, and thereby $P(\mathbb{X}^{(i)}) \neq P(\mathbb{X}^{(j)})$ and $P(\mathbb{Y}^{(i)}) \neq P(\mathbb{Y}^{(j)})$ if $i \neq j$, *without* the task label t provided to the model during inference or evaluation.

A critical problem in continual learning is *catastrophic forgetting* [Kirkpatrick et al., 2017, Li and Hoiem, 2017, Zenke et al., 2017, Aljundi et al., 2018, Diethe et al., 2019, Van de Ven and Tolias, 2019]. As mentioned above, it refers to the fact that the model tends to overfit new training data, forgetting previously acquired knowledge, and thereby degrading the inference performance on old classes. This originates from the imbalance between the data for old and new classes because none or only part of the old data can be retained during fine-tuning [Hou et al., 2019, Wu et al., 2019]. Moreover, the data imbalance also induces bias in the classification layer: old samples are often more likely to be classified as new classes after training [Zhao et al., 2020].

To deal with catastrophic forgetting, researchers have proposed different techniques in continual learning, by striking a balance between the stability (ability to retain knowledge) and plasticity (ability to learn new concepts) of deep learning models. Broadly, continual learning techniques can be categorised into three types [De Lange et al., 2021]: regularisation-based [Kirkpatrick et al., 2017, Li and Hoiem, 2017, Shin et al., 2017, Wu et al., 2019], replay-based [Rebuffi et al., 2017, Chaudhry et al., 2019, Ostapenko et al., 2019, Buzzega et al., 2020], and parameter-isolation methods [Rusu et al., 2016, Serra et al., 2018].

Regularisation-based methods first locate the important model parameters that contribute more to the classification decision by using importance metrics such as fisher information [Kirkpatrick et al., 2017] and output gradients [Aljundi et al., 2018]. An additional term is then added to the loss function to restrict the changes in these parameters. Similarly, self-training and knowledge distillation techniques (as introduced in Section 2.2.4), have also been used to retain knowledge when learning for new tasks [Li and Hoiem, 2017, Zhou et al., 2019]. An extra term is added to the loss function to minimise the difference between the outputs from the new model and those from the old model during incremental training.

Replay-based (also called exemplar-based) approaches involve selecting a small portion of samples from previous tasks (randomly or with herding techniques [Lopez-Paz and Ranzato, 2017, Hayes et al., 2020, Iscen et al., 2020]) and replaying it at later incremental steps. These approaches have shown impressive performance in retaining knowledge while learning new concepts. To correct the bias due to the data imbalance, researchers have proposed to apply normalisation on the weights [Hou et al., 2019, Zhao et al., 2020] or to add an extra correction layer on top of the classification layers [Wu et al., 2019]. State-of-the-art continual learning performance is often achieved using a combination of different techniques [Mittal et al., 2021].

Another challenge in deploying continual learning systems in the real world is the assumption that a large amount of labelled data is available for training for every new task. This prevents continual learning techniques from being widely applicable because collecting high-quality labelled data, as we have seen, is very difficult. This especially applies when data is generated on the fly in mobile sensing. For example, a fitness tracker can obtain raw sensor signals in the background when the user is performing a new exercise, but it does not have access to the ground truth.

To mitigate this problem, continual self-supervised learning (CSSL) approaches, which operate on unlabelled data while dealing with catastrophic forgetting, have been proposed. Early techniques in this area focus on either self-supervised pre-training and later applying supervised continual learning [Gallardo et al., 2021, Caccia and Pineau, 2022] or extending contrastive learning setups to continual learning [Cha et al., 2021, Madaan et al., 2021]. However, these techniques have a narrow focus as they are tailored to specific self-supervised learning architectures. Few recent works [De Lange et al., 2021, Fini et al., 2022] started looking at general frameworks for unsupervised continual learning, where the model learns continually from a stream of unlabelled data using a combination of unsupervised learning techniques and knowledge retention mechanisms, but they did not address how the classifier can be continually trained.

2.3 Research gap

The review of existing works in human activity recognition and different deep learning training paradigms reveals research gaps in the literature and motivates the designs of the training paradigms proposed in this thesis.

Although we have seen success in leveraging self-supervised learning techniques for human activity recognition (see Section 2.2.2), many of these works do not effectively leverage large-scale unlabelled datasets collected in free-living environments. Since the performance of neural networks heavily depends on the data that they are trained on, in Chapter 3 we propose a training strategy that incorporates elements of self-supervised learning and self-training, so that models can be trained on large amounts of data with a high level of diversity. Our proposal is able to make use of large unlabelled datasets efficiently by using self-training with sample filtering, and the performance is further improved with the transformation discrimination task. This proposal shines a light on how well-performing models can be developed in a data-efficient manner, overcoming the limitations of labelled datasets.

We have also seen that previous researchers have looked at how contrastive learning has shown impressive results using unlabelled data (see Section 2.2.3). Motivated by the observation that different body-worn devices observe the same physical activity from different viewpoints, which can be viewed as 'natural transformations', we propose a novel collaborative training pipeline leveraging the contrastive learning setup without the need to manually design transformation functions, which is presented in Chapter 4. The proposal contrasts time-aligned samples from different devices against samples from other timestamps, forming a data-efficient training pipeline that is able to leverage unlabelled sensor data from multiple devices.

Looking beyond the conventional supervised learning setup, where the set of activities is pre-defined, in Chapter 5 we look at how multi-task learning and self-supervised learning techniques can also be used in the context of continual learning (as discussed in Section 2.2.7), where the data distribution changes and new tasks are introduced over time. Building upon previous multi-task learning research (see Section 2.2.6), we investigate how multi-task learning can be used to improve data representations at the initial learning step in Section 5.2. Section 5.3 further builds upon existing self-supervised learning and self-training works in improving feature generalisability, resulting in a unified scheme to combat catastrophic forgetting.

Chapter 3

Improving human activity recognition through self-training with unlabelled data

In this chapter, we present a novel training pipeline that effectively leverages unlabelled mobile sensing datasets to complement small labelled datasets. This addresses the first research gap highlighted in Section 2.3 by combining self-training (as described in Section 2.2.4) to distill knowledge from labelled and unlabelled data, along with multi-task self-supervision (as described in Section 2.2.2 and Section 2.2.6), for robust representation learning. Our semi-supervised approach improves the data efficiency of human activity recognition models by effectively leveraging abundant unlabelled data.

3.1 Motivation and overview

In order to overcome the inherent limitations of labelled datasets highlighted in Section 1.1, we begin by focusing on semi-supervised learning, a training paradigm that leverages abundant unlabelled data to complement labelled data. Its potential for expanding the training data for HAR models has been highlighted in Section 2.2.1. Our aim is to increase the diversity and quantity of training data for HAR models, which in turn improves the generalisability and performance of these models.

Here we introduce *SelfHAR* (see Figure 3.1), a semi-supervised framework that complements supervised training on labelled data, using self-supervised pre-training to leverage the information captured by large-scale unlabelled datasets in HAR tasks. This approach implicitly introduces more diverse sensor data to our training paradigm, therefore offering a unique avenue for performance improvement. Specifically, the approach combines different training techniques to form an efficient and effective training pipeline: teacher-student self-training, which has been covered in Section 2.2.4, allows large amounts of unlabelled

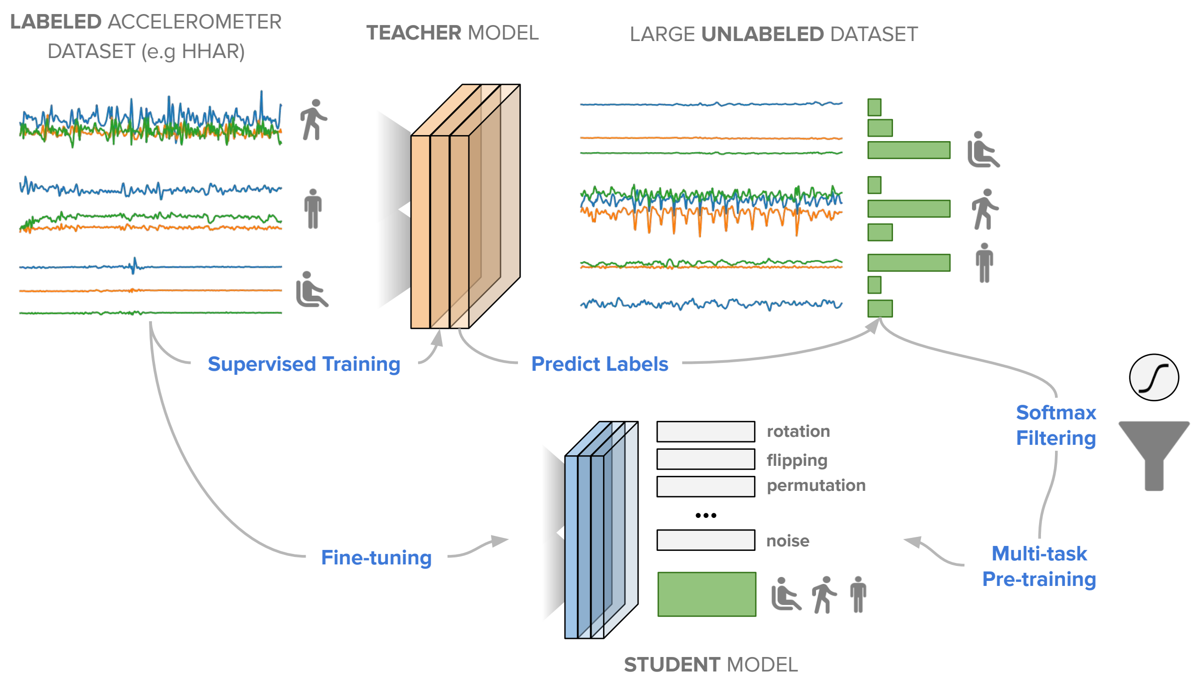


Figure 3.1: Overview of the proposed approach *SelfHAR*. A teacher model distills the knowledge of labelled accelerometer data and then is used to generate pseudo-labels for a large unlabelled dataset. We select only the high-confidence data points from the previous step and train a student model to discriminate signal transformations as well as the activities. Lastly, the ground-truth labels from the training set are used to fine-tune the student model.

data to be efficiently used, and multi-task self-supervised learning, as discussed in Section 2.2.2 and Section 2.2.6, allows the model to be more robust against sensor noises. These methods are able to increase the internal and external diversity of the training data, which allows the model to learn more generalisable features. The labelled datasets are effectively extended with the unlabelled data in this approach. We examine the performance of our method on several datasets collected from a diverse group of participants, utilising different devices and sensors and with different experimental setups. We demonstrate increased performance without increasing model complexity at inference time, which could have important implications for the real-world deployment of machine learning models on resource-constrained mobile devices. Additionally, we evaluated the models in scenarios with limited training data, showing similar performance achieved with 10 times less labelled data compared to the conventional fully supervised approach.

Our work in this chapter makes three major contributions:

1. We present *SelfHAR*, a training paradigm that incorporates the teacher-student setup into self-supervised pre-training. This combination of methods leverages large

unlabelled wearable and mobile sensing datasets more effectively, enabling deep learning models to learn feature extraction and representations in a multi-task setup.

2. We evaluate *SelfHAR* on seven datasets, consisting of different sensor types, populations and protocols. We demonstrate that leveraging unlabelled datasets to increase the device, placement and user diversity leads to a further improvement in performance, outperforming state-of-the-art supervised and semi-supervised methods by up to 12% in F_1 score. This superior performance was attained without increasing the complexity of the machine learning model. With our models, we show that solely using pre-training can only go so far in terms of handling diverse unlabelled datasets.

3. Our proposed approach shows robust results in cases where limited amounts of data are available. An F_1 score of $0.67 - 0.88$ was attained with only 10 labelled samples per activity class, and a similar performance was achieved with up to 10 times less labelled data compared to the conventional fully-supervised approach.

3.2 Approach

In this section, we describe *SelfHAR*, our semi-supervised training framework for HAR, which incorporates self-training and self-supervised learning for leveraging unlabelled datasets to improve the diversity of training data.

3.2.1 Problem statement

We consider the problem of HAR, defined as follows (inspired by the definition provided by [Lara and Labrador, 2013]): given a set $\mathbb{L}$ of time series data, a set $\mathbb{A} = \{a_i, \dots, a_n\}$ of n activity labels, and a set $\mathbb{W} = \{w_1, \dots, w_m\}$ of m time windows defining start and end-points of time periods, the task is to find a mapping function f such that $f(\mathbb{L}, w_i)$ outputs an activity label a_i, where a_i should represent the actual activity performed by the subject during w_i. In addition to the labelled dataset $\mathbb{L}$, we consider scenarios where there is a complementary dataset $\mathbb{U}$, collected from similar sensors as $\mathbb{L}$ but without labels. We aim to train models such that improvement in recognition accuracy can be achieved by using $\mathbb{L} \cup \mathbb{U}$ over using only $\mathbb{L}$ (i.e., by leveraging an unlabelled dataset). The evaluation is conducted by asking the models to predict the activity labels of an unseen subset of $\mathbb{L}$.

3.2.2 The SelfHAR training pipeline

Overview

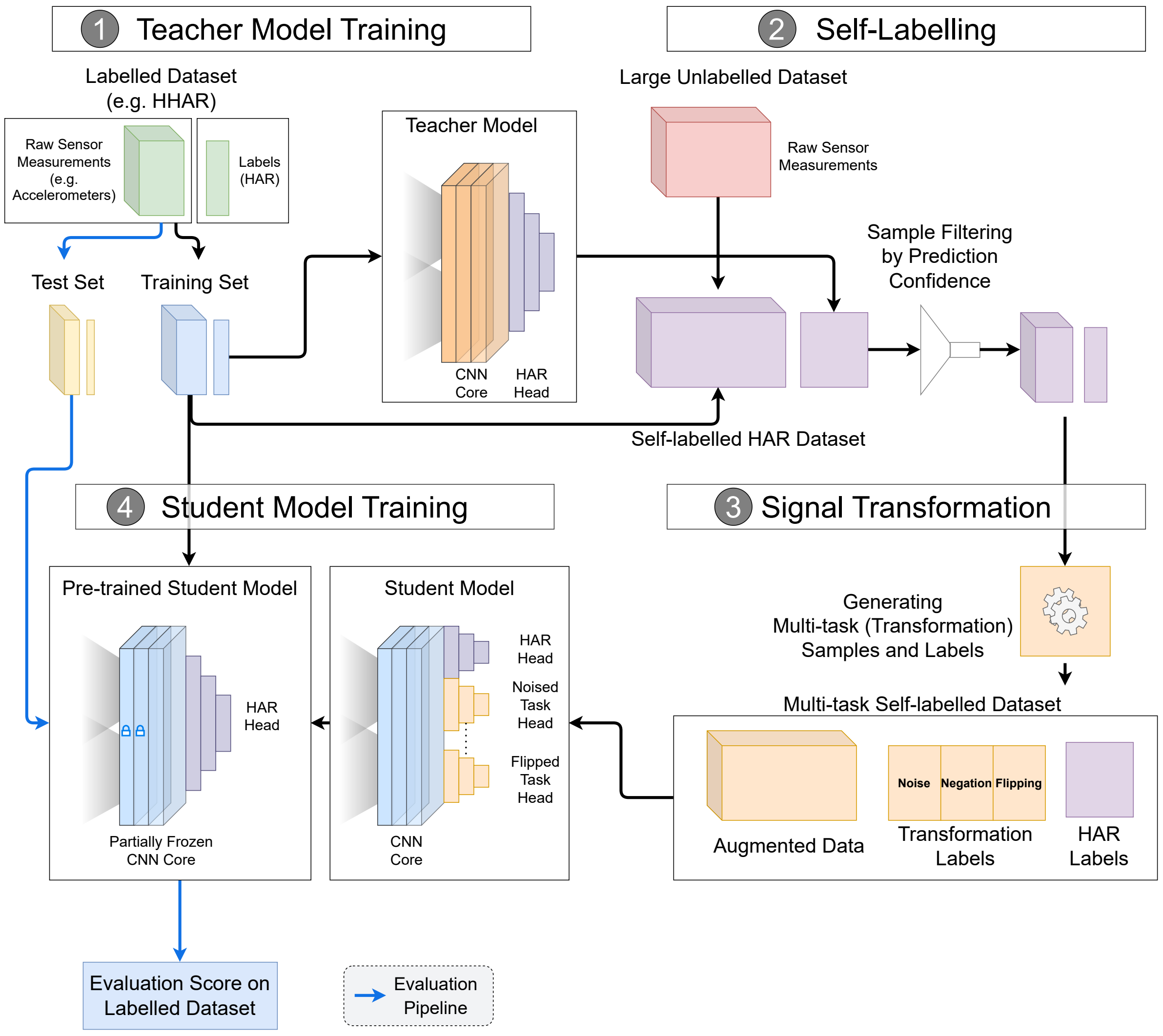


Figure 3.2: Detailed schematic of the proposed pipeline *SelfHAR*. The pipeline adopts a teacher-student setup, where a teacher model is first trained and distills the knowledge of labelled data by labelling a large unlabelled dataset. High-confidence data points from the previous step are selected and then augmented through signal transformation to form a multi-task training dataset. The student model is first pre-trained to discriminate signal transformations as well as the activities, and the ground-truth labels from the training set are then used to fine-tune it. Evaluation is then performed on an unseen subset of the target dataset.

Algorithm 1 *SelfHAR* - Combined semi-supervised learning

Input: Labelled dataset $\mathbb{L}$, unlabelled dataset $\mathbb{U}$, Transformation functions $\mathbb{H}$, Activity classes $\mathbb{A}$, Number of teacher training epochs E_T, Number of student pre-training epochs E_P, Number of student fine-tuning epochs E_S, Confidence threshold C, Maximum number of samples per activity class K

Output: A Trained HAR Model `model_student`

```
# Teacher model training
model_teacher = new_har_model()
for epoch in [1 ... ET]:
    minibatch_train(model_teacher, dataset=L, loss=CrossEntropy(),
                    optimizer=GradientDescent())

# Use the teacher model to generate pseudo-labels
V = empty_dataset()
V.X = combine_datasets(L.X, U)
V.Y = model_teacher.predict(V.X)

# Select the most confident samples from the dataset
S = []
for activity_class in A:
    V_sorted = sort(V, key = V.Y.get_entry(activity_class))
    V_filtered = filter(V_sorted,
                        condition = V_sorted.Y.get_entry(activity_class) >= C)
    V_selected = V_filtered.get_first_n_entries(n = K)
    S.append(V_selected)

# Augment the data and construct a multi-task training dataset
L_prime = []
for (signals, har_label) in S:
    y = [0, 0, ..., 0]
    L_prime.append((signals, har_label, copy(y))
    for transformation in H:
        y[transformation] = 1 # To mark the transformation performed on the sample
        L_prime.append( (transformation(signals), har_label, copy(y) )
        y[transformation] = 0

# Student pre-training
model_student = new_multitask_model()
for epoch in [1 ... EP]:
    minibatch_train(model_student, dataset=L_prime, loss=CrossEntropy() +
                    BinaryTransformationLoss(), optimizer=GradientDescent())

# Student fine-tuning
for layer in model_student.get_core_layers():
    freeze_weights(layer)
for epoch in [1 ... ES]:
    minibatch_train(model_student, dataset=L, loss=CrossEntropy(),
                    optimizer=GradientDescent())
```

The proposed training pipeline combines self-training and self-supervised pre-training into a multi-step training pipeline (see Figure 3.2), which contains different elements of the conventional training pipeline of self-training and self-supervised learning techniques (as discussed in Section 2.2.2 and Section 2.2.4):

1. We employ a self-training paradigm where a teacher model is first trained on the labelled dataset $\mathbb{L}$.
2. The labels from the labelled dataset $\mathbb{L}$ are removed and mixed with the unlabelled dataset $\mathbb{U}$, forming a mixed dataset $\mathbb{V}$ without labels.
3. The mixed dataset $\mathbb{V}$ is then labelled by running the teacher model, where these labelled samples are filtered by a minimum confidence threshold C and the top K samples for each class is selected, forming an intermediate dataset $\mathbb{S}$.
4. The selected samples $\mathbb{S}$ are augmented by the eight signal transformations used in [Saeed et al., 2019]. This generates augmented sensor data and transformation labels, forming the self-supervised dataset $\mathbb{L}'$.
5. A student model is pre-trained with the self-supervised dataset $\mathbb{L}'$ in a multi-task learning setting.
6. The student model is then fine-tuned on the initial labelled dataset $\mathbb{L}$ and evaluated.

This pipeline is designed to increase the diversity of the training data internally through augmentation and externally through the use of unlabelled data, which allows the model to learn more generalisable features. The algorithm of the training pipeline is outlined by Algorithm 1.

We describe the individual steps in our training pipeline in detail in the following sections.

Teacher model training

First, we employ a knowledge distillation paradigm [Hinton et al., 2015] where a teacher model is trained on the labelled dataset $\mathbb{L}$ following conventional deep learning training: a multi-class activity classification loss function is used with gradient descent and regularisation to train the teacher model iteratively. The main goal of this step is to train a model to generate activity labels for self-training. The cross-entropy loss for multi-class classification (which has been introduced in Section 2.1.3) is used for training and can be defined as:

$$\mathcal{L}_{\text{CE}} = -\frac{1}{|\mathbb{L}|}\sum_{d\in\mathbb{L}}\sum_{a\in\mathbb{A}} y_{d,a}\log(\{M_\theta(d)\}_a) \tag{3.1}$$

Where $\mathbb{L}$ is the labelled dataset, $\mathbb{A}$ is the set of activity labels (activity classes), M_θ is the model parameterised by θ, $y_{d,a}$ is 1 if the ground-truth activity label is a in window d or 0 otherwise, and $\{M_\theta(d)\}_a$ is the probability of window d having label a as predicted by the model.

Samples labelling and selection

After the teacher model is trained, activity labels for the mixed dataset $\mathbb{V}$ are generated by running the teacher model on the sensor signals. For each activity class, samples are ranked by the confidence in that particular class, and the top K samples out of those which have a softmax score of at least C in that class are selected. The probabilistic labels are kept during selection and used for training the student model. This step is to select the most confident samples in order to limit the labelling noise in each class [Yalniz et al., 2019] while curating a large labelled dataset. The confidence score threshold is used to make sure samples that the teacher model is uncertain about do not get selected.

Data augmentation and further labelling

Once the samples with the highest softmax scores are selected to form $\mathbb{S}$, eight signal transformation functions, as used in [Um et al., 2017] and [Saeed et al., 2019], are used to augment the dataset: adding random noise, scaling using a random scalar, applying a random 3D rotation, inverting the signals, reversing the direction of time, randomly scrambling sections of the signal, stretching and warping the time-series, and shuffling the different channels. These tasks are selected because mobile devices can be put in very different environments during use. For example, smartphones can be placed in a pocket, a bag, or held in a user's hand while performing different activities. Variations among devices and sensors are also common, which causes the signal to exhibit different forms of noise, including a change in orientation or signal magnitude. It is our goal to enable the models to be resilient against such noises and to learn invariants within the signals.

Using the transformation functions, each sample in the dataset $\mathbb{S}$ is augmented to a total of nine different versions consisting of the original and one for each transformation. These augmented samples come with eight binary transformation labels, each indicating whether a particular transformation has been applied. There is not any mixing of two or more transformations applied to a sample simultaneously to avoid confusion and for ease of evaluation. The HAR activity labels are duplicated from the original signal, forming a multi-task self-training dataset $\mathbb{L}'$.

Student model training

The student model is a multi-task prediction model with nine prediction tasks: eight binary transformation discrimination tasks and one multi-class activity classification. It is trained in a supervised manner using the dataset $\mathbb{L}'$. A combined loss function is formed by summing individual losses for each task, allowing the model to learn all tasks simultaneously. After training, the early layers of the student model are fixed (frozen), and the activity classification branch is extracted and fine-tuned using the original dataset $\mathbb{L}$. This is similar to transfer learning setups where a part of the network is frozen and the common layers are transferred to a new model. Through this process, the models are aligned to the target HAR task, while useful feature extraction layers early in the model are retained. Early stopping is also used in both pre-training and fine-tuning to reduce over-fitting.

The classification loss for the HAR task, the classification loss for the transformation discrimination task, and the combined loss function for pre-training are as follows:

$$\mathcal{L}_{\text{CE}}^{\text{HAR}} = -\frac{1}{|\mathbb{L}'|} \sum_{d \in \mathbb{L}'} \sum_{a \in \mathbb{A}} y_{d,a}^{\text{HAR}} \log(\{M_{\theta'}(d)\}_a^{\text{HAR}}) \tag{3.2}$$

$$\mathcal{L}_{\text{CE}}^{\text{TD}} = \sum_{h \in \mathbb{H}} \left[-\frac{1}{|\mathbb{L}'|} \sum_{d \in \mathbb{L}'} \sum_{a \in \{\text{True,False}\}} y_{d,a,t}^{\text{TD}} \log(\{M_{\theta'}(d)\}_{a,h}^{\text{TD}}) \right] \tag{3.3}$$

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{CE}}^{\text{HAR}} + \mathcal{L}_{\text{CE}}^{\text{TD}} + \beta(||\theta'||^2) \tag{3.4}$$

Where $\mathcal{L}_{\text{CE}}^{\text{HAR}}$ represents the loss function for the HAR task, $\mathcal{L}_{\text{CE}}^{\text{TD}}$ is the loss function for the transformation discrimination task, $\mathbb{L}'$ is the combined self-supervised training dataset, $\mathbb{H}$ is the set of transformation functions, $\{M_{\theta'}(d)\}_a$ is the confidence of window d having label a predicted by the student model $M_{\theta'}$, $y_{d,a,h}^{\text{TD}}$ is 1 if the signal d was transformed from the original using the function h or 0 otherwise. β is the coefficient of the L2 regularisation which is applied to the model parameters θ'.

3.2.3 Configurations of the SelfHAR pipeline

From Figure 3.2 and the descriptions in Section 3.2.2, the *SelfHAR* training pipeline can be decomposed into 4 components: (1) Teacher model training, (2) Self-labelling, (3) Signal transformation and (4) Student model training. In addition to this configuration, the components can be arranged to form other pipelines. In particular, the transformation discrimination task can be used on its own for self-supervised training of the teacher model, instead of combined into the student training setup (component 3). As using the transformation discrimination task for pre-training the teacher is performed before fine-tuning the teacher (component 1), it will be denoted component 0 in our study (see Figure 3.3 for a visualisation of the components).

A total of five different configurations were explored in this study: (1) fully supervised (using component 1 only), (2) transformation discrimination training (using components

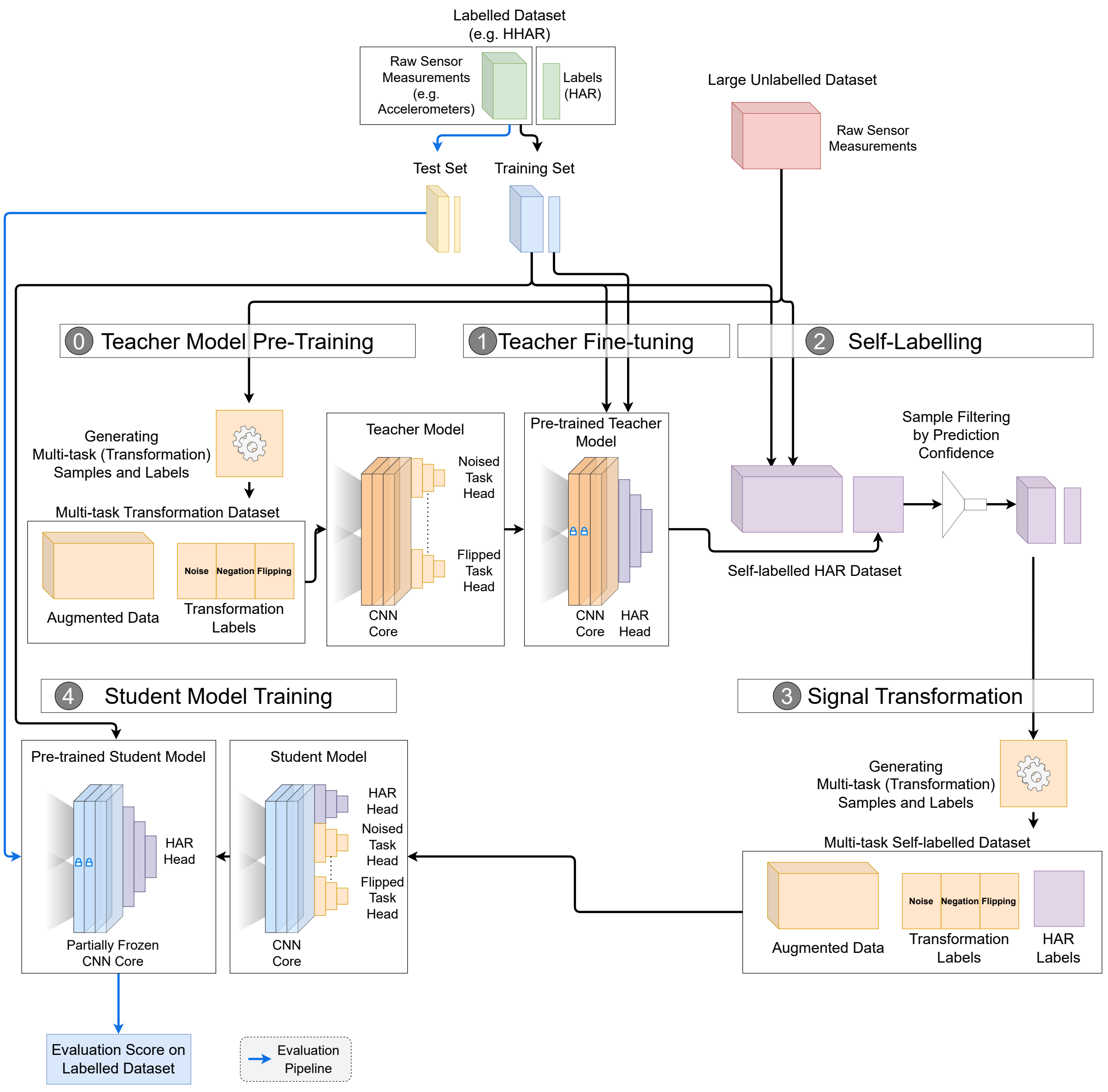

Labelled Dataset (e.g. HHAR)
Raw Sensor Measurements (e.g. Accelerometers)
Labels (HAR)
Large Unlabelled Dataset
Raw Sensor Measurements
Test Set
Training Set
0 Teacher Model Pre-Training
1 Teacher Fine-tuning
2 Self-Labelling
Generating Multi-task (Transformation) Samples and Labels
Multi-task Transformation Dataset
Augmented Data
Noise
Negation
Flipping
Transformation Labels
Teacher Model
Noised Task Head
Flipped Task Head
CNN Core
Pre-trained Teacher Model
CNN Core
HAR Head
Sample Filtering by Prediction Confidence
Self-labelled HAR Dataset
4 Student Model Training
3 Signal Transformation
Pre-trained Student Model
HAR Head
Partially Frozen CNN Core
Student Model
HAR Head
Noised Task Head
Flipped Task Head
CNN Core
Generating Multi-task (Transformation) Samples and Labels
Multi-task Self-labelled Dataset
Augmented Data
Noise
Negation
Flipping
Transformation Labels
HAR Labels
Evaluation Score on Labelled Dataset
Evaluation Pipeline

Figure 3.3: The extended SelfHAR pipeline. Five different configurations of the proposed pipeline were evaluated in our ablation studies.

0 and 1), (3) self-training (using components 1, 2 and 4), (4) transformation knowledge distillation (using components 0, 1, 2 and 4) and (5) *SelfHAR*, which follows our approach proposed above (using components 1, 2, 3 and 4). In one of our ablation studies (see Section 3.4.5), we further experimented with variations of the transformation discrimination and *SelfHAR* pipelines without the use of an unlabelled dataset.

The fully supervised training pipeline follows conventional supervised training, in which only one model is trained through supervised learning. The transformation discrimination pipeline pre-trains models with the transformation discrimination task using the unlabelled dataset. After pre-training, the convolutional core was transferred and a randomly initialised activity recognition head was attached to the convolutional core. The final layer of the convolutional core and the new activity recognition layers were then fine-tuned on the labelled dataset. This follows the method proposed by [Saeed et al., 2019] closely.

On the other hand, the self-training pipeline consists of a pure teacher-student setup without signal transformation. The pipeline follows the *SelfHAR* setup closely, but the self-labelled HAR dataset was not augmented with signal transformation and the student model was pre-trained purely on the HAR labels generated by the teacher model. The student was similarly fine-tuned with the labelled dataset at the end and evaluated with the same test set. The transformation knowledge distillation pipeline enhances the self-training-only pipeline by pre-training the teacher model with the task of transformation discrimination.

It is important to note that these models share the same neural network architecture, and the resulting prediction models share the same complexity. An ablation study is performed to evaluate these different pipelines in this study (see Section 3.4.2).

3.3 Experimental protocols

In this section, we describe the datasets that we used to evaluate our proposed approach as well as the associated experimental protocols.

3.3.1 Datasets

Eight datasets were used in this study, where seven of them are labelled and one of them is unlabelled (see Table 3.1). A description of each dataset used in our work is provided below.

HHAR

The Heterogeneity Activity Recognition dataset (HHAR) [Stisen et al., 2015] is a dataset collected to investigate the impact of heterogeneous devices on activity recognition performance, with a focus on the variations between different sensors, devices and workloads. The data was collected from 9 participants (aged between 25 and 30) who performed 6 common daily activities: biking, sitting, standing, walking, walking upstairs and walking downstairs. The data from accelerometers and gyroscopes embedded in 8 smartphones placed around the users' waist and 4 smartwatches around the users' arms with varying sampling frequencies were collected. Two of each of the following devices were used: LG Nexus 4 (200 Hz), Samsung Galaxy S Plus (50 Hz), Samsung Galaxy S3 (150 Hz), Samsung Galaxy S3 mini (100 Hz), LG G (200 Hz) and Samsung Galaxy Wear (100 Hz). The

Table 3.1: An overview of datasets used in this study. A total of eight datasets were used in this study with different sizes, numbers of users, activities and device placements.

Dataset	Devices position	Users	Activities (labels)		Data Samples Used
HHAR [Stisen et al., 2015]	Waist and Arm	9	6	(biking, sitting, standing, walking, walking upstairs, walking downstairs)	56383
MotionSense [Malekzadeh et al., 2018]	Front Pocket (Trousers)	24	6	(walking downstairs, walking upstairs, walking, jogging, sitting, standing)	6630
MobiAct [Chatzaki et al., 2016]	Front Pocket (Trousers)	66	11	(standing, walking, jogging, jumping, walking upstairs, walking downstairs, standing to sitting on a chair (activity transition), sitting on a chair, sitting to standing, stepping into a car, stepping out of a car)	57995
PAAS [van Hees et al., 2013]	Wrist	28	11	(sitting, standing, walking, walking upstairs, walking downstairs, lying, cycling, home activities, office activities, personal care, shopping)	32558
UniMiB SHAR [Micucci et al., 2017]	Front Pocket (Trousers)	30	9	(standing up from lying, lying down to standing, standing up from sitting, running, sitting down, going downstairs, going upstairs, walking, jumping)	1551
UCI HAR [Anguita et al., 2013]	Waist	30	6	(walking, walking upstairs, walking downstairs, sitting, standing, lying down)	2543
WISDM [Kwapisz et al., 2011]	Front Pocket (Trousers)	29	6	(walking, jogging, sitting, standing, walking upstairs and walking downstairs)	5435
Fenland [Lotta et al., 2018]	Wrist	2096	N/A (Unlabelled)		168687 (Subset)

participants were asked to perform the 6 activities for 5 minutes each, to ensure an equal data distribution among the different classes. Only the sensor signals from the triaxial accelerometer of the smartphones were used in this study due to the need for compatibility among datasets.

MotionSense

The MotionSense [Malekzadeh et al., 2018] dataset was collected for the development of privacy-preserving sensor data transmission systems, where the data was used to test whether personal attribute fingerprints could be extracted from the data. 24 subjects (14 men and 10 women) were recruited. The body mass of the subjects ranged between 48 kg and 102 kg, their height ranged between 161 cm and 190 cm, and their age ranged between 18 and 46. The acceleration, gravity and attitude (pitch, roll, yaw) were collected at 50 Hz from an iPhone 6s placed in the trousers' front pocket, while performing 6 activities: walking downstairs, walking upstairs, walking, jogging, sitting, and standing. 15 trials were conducted in the same environment and condition around the Mile End campus of the Queen Mary University of London, with each trial lasting between 30 seconds and 3 minutes. In our study, only signals from the accelerometer embedded in the iPhone 6s were used.

MobiAct

The MobiAct dataset (second release) [Chatzaki et al., 2016] consists of accelerometer, gyroscope and orientation readings from a Samsung Galaxy S3 (LSM330DLC inertial module at 20 Hz) for the purpose of developing a combined system which can perform HAR and fall detection. 66 participants (51 men and 15 women), of age 20-47, height 160-193 cm and body mass 50-120kg, were recruited to perform 12 types of activities of daily living (ADLs) and 4 types of falls. Fall-related activities, including sitting, stepping in a car and jumping, were selected in order to test the robustness of the developed system in detecting actual falls. A total of 3200 trials were conducted with the participants performing the ADLs and acting out 5 different daily living scenarios. The smartphone was placed freely in the participant's pockets without specifying orientation, with the exception of fall detection where the smartphone was placed in the pocket on the opposite side of where the participant would be landing.

For our work, only data unrelated to falls was used as we were interested in activity classification. These included 11 types of activities: standing, walking, jogging, jumping, walking upstairs, walking downstairs, standing to sitting on a chair (activity transition), sitting on a chair, sitting to standing, stepping into a car and stepping out of a car.

PAAS

The Physical Activity Annotation Study (PAAS) was established to develop algorithms for physical activity type classification based on a single sensor location while developing a multi-sensor system to serve as a gold standard for physical activity classification in future studies. A detailed description of the study, the sensors used and the activity breakdown has been previously published [van Hees et al., 2013]. The study recruited 28 healthy adult participants (15 women, 13 men) who were tested at the Institute of Metabolic Science at Addenbrookes Hospital in Cambridge, UK. For the PAAS study, participants wore nine triaxial raw accelerometers (GENEA). Additionally, four other sensors were positioned on the participant's body, including one Actigraph GT3X (Actigraph), one Sensewear (Bodymedia), and two ActiWare Cardio monitors (CamNtech). None of the monitors was obstructive of normal body movement. All participants performed a protocol that included 60 activities, which aimed to capture the majority of activities that an individual who works at an office does during a normal day. An audio recording was used to provide participants with activity instructions for the different sections of the protocol, ensuring that all participants' activities were recorded for the same amount of time. The full activity breakdown has also been described previously in detail [van Hees et al., 2013]. The protocol aimed to be representative of occupational activity and capture the sedentary behaviours associated with most desk jobs. Participants also wore a subset of the sensors in free-living conditions after the protocol, but this data was not utilised for our experiments.

For our analysis, due to the relatively small size of the dataset and for simplicity purposes, activities were grouped into 11 categories: sitting, standing, walking, walking upstairs, walking downstairs, lying, cycling, home activities, office activities, personal care and shopping. In our experiments, we only used the triaxial accelerometer (GENEA, previously described) placed on the non-dominant wrist of every participant. This sensor has a dynamic range of $\pm 6g$ and is sampled at 80Hz. Timestamps were also stored for the full recording.

UniMiB SHAR

The UniMiB SHAR dataset [Micucci et al., 2017] was also collected for HAR and fall detection. The data was collected from 30 healthy adults (24 women and 6 men), of age 18-60, height 160-190 cm and body mass 50-82 kg. They were asked to perform 9 types of ADLs and 8 types of falls, which were selected given their popularity in other public datasets. Acceleration data from a Samsung Galaxy Nexus I9250 (with Bosh BMA220 acceleration sensor) placed in the front trouser pockets were collected at 50 Hz, and audio recordings were also collected for data annotation. Four different sequences were designed to allow the participants to perform with ease. Among the trials, the smartphones were placed in the left trouser pocket for half of the time, and in the right one for the other half of the time. Written informed consent was obtained from the participants, in accordance with the World Medical Association (WMA) Declaration of Helsinki. Data for the 9 classes of ADLs were used in this study: standing up from lying, lying down to standing, standing up from sitting, running, sitting down, going downstairs, going upstairs, walking, and jumping.

UCI HAR

The UCI HAR dataset [Anguita et al., 2013] was established for developing HAR algorithms using smartphones. The data came from 30 volunteers (aged 19-48) performing 6 different activities: walking, walking upstairs, walking downstairs, sitting, standing and lying down. The volunteers followed a protocol that lasted 192 seconds, and performed it twice, with a Samsung S II mounted on the left side of a belt around the waist for the first time, and the same phone placed on a location on the belt where the user preferred during the second time. Sensor signals from the accelerometer and the gyroscope were collected at 50 Hz.

WISDM

The WISDM (Wireless Sensor Data Mining) project [Kwapisz et al., 2011] was an early study set to explore issues in obtaining sensor data from mobile devices. The data was collected from 29 volunteers performing 6 activities of daily living which are common in daily routines: walking, jogging, sitting, standing, walking upstairs and walking downstairs. The participants carried a smartphone (Nexus One, HTC Hero, or Motorola Backflip) in one of their trousers' front pockets and performed a varying number of times for each activity. The data from the accelerometer embedded in the smartphone was collected at a sampling frequency of 20 Hz. An approval from the Fordham University IRB (Institutional Review Board) was obtained before the data collection.

Fenland Study

The Fenland Study is a prospective cohort study of 12,435 men and women aged 35-65 with a primary objective of identifying behavioural, environmental, and genetic causes of obesity and type-2 diabetes. The study recruited participants from three different sites, excluding participants with clinically diagnosed diabetes mellitus, inability to walk unaided, terminal illness, clinically diagnosed psychotic disorder, pregnancy, or lactation. The Fenland study has been previously described in detail by [Lotta et al., 2018]. After a baseline clinic visit, a subsample of 2,100 participants was asked to wear a wrist accelerometer (GeneActiv) on the non-dominant wrist. This device recorded triaxial acceleration at 60 Hz. Participants were instructed to wear both waterproof monitors continuously for six full days and nights during free-living conditions, including during showering and while they were sleeping. This subsample of participants constitutes the sampling frame for the current analyses. Further, we obtained physical activity energy expenditure (PAEE) from the triaxial accelerometry sensor using a method described and validated in a previous study [White et al., 2016]. This information was then used to obtain time-series data of metabolic equivalents (METs) and label them into sedentary (≤ 0.5 METs), light physical activity (LPA) (0.5-3 METs) and moderate to vigorous physical activity (MVPA) (≥ 3 METs) following the conversion that 1 standard MET is 71 $\text{J}\cdot\text{min}^{-1}\cdot\text{kg}^{-1}$.

3.3.2 Data preparation

Minimal pre-processing was applied to the raw sensor datasets. The data was first normalised by applying z-normalisation using the mean and standard deviation of the training data for each of the sensor channels. Each dataset was then segmented into sliding windows with size 400×3, where 400 represents the number of timestamps and 3 represents the number of channels of a triaxial accelerometer. The sliding windows shared 50% overlap, where the next window had a starting-time shifted 200 timestamps to the future. Data from $20\% - 25\%$ of users from each labelled dataset was kept unseen to the models and used as the test set in order to evaluate the generalisability of the models. There was no re-sampling performed to standardise the sampling frequencies of the datasets. The same configurations and pre-processing procedures were kept across different datasets because we wanted to highlight that the performance achieved was a result of the training paradigm but not because of the differences in configurations or pre-processing procedures. Furthermore, a similar pre-processing protocol was also adopted in previous work [Saeed et al., 2019] and, to make for a fair comparison, we followed similar pre-processing steps.

3.3.3 Experimental setup

Standard evaluation

To evaluate the training pipeline irrespective of the neural network architecture, and to draw direct comparisons with previous work, a simple model architecture, TPN, as introduced in [Saeed et al., 2019], was adopted. The model consisted of a temporal convolutional core with task-specific heads attached to it. The core consisted of three temporal (1D) convolutional layers, with 32, 64 and 96 filters and kernel sizes of 24, 16, and 8 respectively, all with stride 1. The ReLU activation function was used with L2 regularisation with a factor of 0.0001 for the weights. Between every pair of layers, a drop-out layer with a rate of 0.1 was used. A global 1D maximum pooling layer was connected to the last convolutional layer, and this formed the convolutional core.

Depending on the setup, different task-specific heads were used. For transformation discrimination tasks, the last layer of the convolutional core was connected to a fully connected (dense) layer of 256 hidden units, activated by ReLU. Following this was a fully connected layer of 1 unit, activated by the sigmoid function, which formed the output layer for one transformation discrimination task. Binary cross-entropy was used as the loss function. For activity recognition, a fully connected layer of 1024 hidden units was connected to the convolutional core, and activated by ReLU. An output prediction layer with the number of units equal to the number of activities was then attached with full connections to the previous dense layer and activated by the softmax function with categorical cross-entropy as the loss function. It is important to note that when a model was trained on a multi-task dataset, the weights of the convolutional core were shared among all tasks. The losses of different tasks were summed with equal weightings during multi-task training. The overall architecture followed the common setup of a multi-task convolutional neural network.

During training, the Adam optimiser [Kingma and Ba, 2015] was used with the decay rate for the first moment being 0.9 the decay rate for the second moment being 0.999 (default settings), and a learning rate of 0.0003. The model was trained for a total of 30 epochs with early stopping, where the model with the lowest validation loss was picked as the final model to mitigate over-fitting. A confidence threshold C of 0.5 was used to filter self-labelled samples in order to ensure plurality (which reflects high confidence), and at most the top $K = 10000$ samples for each class were chosen, maximising the size of the dataset while balancing the distribution among different classes.

During fine-tuning, the weights of the first two layers of the convolutional core were fixed (frozen), and only the weights of the third convolutional layer and subsequent layers were tuned. This design is to allow the model to fine-tune to the final target dataset while retaining previously learned knowledge. This was shown to be the most effective setup in previous work [Saeed et al., 2019].

Linear evaluation

In addition to the standard evaluation protocol mentioned above, the linear evaluation protocol, which is commonly adopted in semi-supervised learning techniques [Zhang et al., 2016, Oord et al., 2018, Bachman et al., 2019, Chen et al., 2020a], was also used in this study. Under this evaluation protocol, after the model has been pre-trained, the convolutional core is completely frozen, and the output is directly connected to a fully-connected layer followed by the softmax function. No additional non-linearity is introduced for the new layer, and the score of the resulting classifier is used as a proxy for the quality of representations extracted by the convolutional core.

Similar to the classification head in standard evaluation, the linear layer was trained on the categorical cross-entropy loss with the Adam optimiser with the same parameters. The weights for the layer were randomly drawn from a normal distribution with the mean being 0 and the standard deviation being 0.01.

Other baseline algorithms

Apart from deep-learning algorithms, two additional semi-supervised learning algorithms, *En-Co-training* [Guan et al., 2007] and *Sparse-Encoding* [Bhattacharya et al., 2014], were used for baseline comparison in our study.

Following the standard protocols proposed in the original work, we used an ensemble of a decision tree, a Naïve Bayes classifier and a 3-nearest neighbour classifier for *En-Co-training* [Guan et al., 2007]. Eight statistical features: (1) mean, (2) correlation between axes, (3) interquartile range (4) mean absolute deviation, (5) root mean square, (6) standard deviation, (7) variance and (8) spectral energy for each axis, as suggested by [Yang et al., 2007], were extracted for each window. The pool size was set to be one-tenth of the number of samples in the unlabelled dataset, and the models were trained for 20 iterations.

Similarly, for *Sparse-Encoding*, a dictionary of 512 basis vectors were learned from the unlabelled data during training [Bhattacharya et al., 2014]. The basis vectors were then separated into 52 clusters, and the lower 10-percentile of vectors in each of the clusters which had low empirical entropy were discarded. A support vector machine with the radial basis function kernel was then trained using grid-search cross-validation.

3.4 Evaluation and discussion

In this section, several evaluation setups were adopted to test the performance of the proposed method in different settings. An in-depth comparison against baseline algorithms is given in Section 3.4.1, with a set of ablation studies to evaluate the effectiveness of the combination of pre-training tasks given in Section 3.4.2. Experiments on the effect of training with limited availability of data are reported in Section 3.4.3. A visualisation of the extracted features is given in Section 3.4.4, followed by a set of ablation studies on the use of unlabelled data in Section 3.4.5, and finally a case study on the effect of different distributions of unlabelled data on the PAAS dataset in Section 3.4.6.

3.4.1 Comparison against baseline algorithms

Overview

We tested the performance of the models using training data from $75\% - 80\%$ of users, with testing data coming from the remaining set of users. The activities that the models were tested on were dataset-dependent. We first built the baseline for comparison using models trained in a fully supervised manner. Results from three other semi-supervised learning algorithms, *En-Co-Training* [Guan et al., 2007], *Sparse-Encoding* [Bhattacharya et al., 2014], and transformation discrimination pre-training [Saeed et al., 2019] were also

Table 3.2: Comparison of classification performance between *SelfHAR* and other HAR algorithms. Weighted F_1 scores were used to benchmark seven different labelled HAR datasets, where *SelfHAR* outperformed the other algorithms in almost all cases. MobiAct was trained with HHAR as the unlabelled dataset.

Dataset	Fully Supervised (Baseline)	En-Co-Training [Guan et al., 2007]	Sparse-Coding [Bhattacharya et al., 2014]	Transformation Discrimination (TD) [Saeed et al., 2019]	*SelfHAR*
HHAR	0.7282 [0.7241, 0.7378]	0.6505 [0.6377, 0.6641] (−7.77%)	0.4140 [0.3517, 0.5019] (−31.42%)	**0.7961 [0.7905, 0.8021] (+6.79%)**	0.7846 [0.7794, 0.7917] (+5.64%)
MotionSense	0.9275 [0.9144, 0.9404]	0.7080 [0.6765, 0.7395] (−21.95%)	0.8015 [0.7735, 0.8278] (−12.60%)	0.9295 [0.9173, 0.9420] (+0.20%)	**0.9631 [0.9530, 0.9726] (+3.56%)**
MobiAct	0.9098 [0.9038, 0.9140]	0.8512 [0.8439, 0.8584] (−5.86%)	0.7848 [0.7616, 0.8097] (−12.50%)	0.8922 [0.8852, 0.8965] (−1.76%)	**0.9355 [0.9305, 0.9394] (+2.57%)**
PAAS	0.6088 [0.5985, 0.6222]	0.5977 [0.5811, 0.6138] (−1.11%)	0.5338 [0.5083, 0.5611] (−7.50%)	0.6851 [0.6741, 0.6967] (+7.63%)	**0.7022 [0.6903, 0.7125] (+9.34%)**
UniMiB	0.7432 [0.6914, 0.7927]	0.6596 [0.6002, 0.7145] (−8.36%)	0.5085 [0.4290, 0.5847] (−23.47%)	0.8098 [0.7600, 0.8516] (+6.66%)	**0.8731 [0.8350, 0.9094] (+12.99%)**
UCI HAR	0.9051 [0.8826, 0.9253]	0.7926 [0.7590, 0.8244] (−11.25%)	0.7620 [0.7260, 0.8005] (−14.31%)	0.9053 [0.8832, 0.9275] (+0.02%)	**0.9135 [0.8929, 0.9354] (+0.84%)**
WISDM	0.8906 [0.8732, 0.9083]	0.6989 [0.6706, 0.7259] (−19.17%)	0.6406 [0.5954, 0.6784] (−25.00%)	0.8948 [0.8780, 0.9136] (+0.42%)	**0.9081 [0.8921, 0.9243] (+1.75%)**

reported. Weighted F_1 scores were used as the main evaluation metric, which allowed us to demonstrate the performance of the models as reflected in different application scenarios captured by a wide range of datasets. Other performance metrics and visualisations were also used in the ablation studies in later subsections. Five independent runs with different model initialisations were performed for each setting in order to mitigate the effect of model initialisation on performance. The average and the 95% bootstrap confidence level of the metrics were reported. They were obtained by re-sampling the test set for 1000 iterations and evaluating the models on them, and the 2.5% and 97.5% percentiles of the evaluation metrics from the models were used to form the confidence intervals.

Two different datasets were used as unlabelled datasets: MobiAct and HHAR. All experiments were performed with MobiAct as the unlabelled dataset, except in the case of evaluating on MobiAct itself, the HHAR dataset was used as the unlabelled dataset instead.

Table 3.2 shows the mean and bootstrap confidence intervals of weighted F_1 scores attained by our proposed pipeline and other training algorithms on the seven target datasets

(HHAR, MotionSense, MobiAct, PAAS, UniMiB SHAR, UCI HAR, and WISDM) over five independent runs. Further, as reported in this table, our proposed method consistently outperformed the fully supervised training baseline, with performance gain up to 12.99% in weighted F_1 score on the UniMiB SHAR dataset over five independent runs. The improvements were statistically significant in HHAR, MotionSense, MobiAct, PAAS, and UniMiB SHAR, where the 95% confidence intervals have no overlap between SelfHAR and the fully supervised training baseline.

When compared against other semi-supervised training algorithms, deep-learning algorithms consistently outperformed the others, with performances lower than the baseline reported by *En-Co-Training* [Guan et al., 2007] and *Sparse-Encoding* [Bhattacharya et al., 2014]. This validated the ability of deep-learning algorithms to recognise complex patterns in high-dimensional data.

Our proposed pipeline outperformed the transformation discrimination algorithm in six out of seven datasets. Statistically significant improvements could be observed in some of the datasets, including MotionSense, MobiAct, PAAS, and UniMiB, in which there was very little to no overlap between the confidence intervals of *SelfHAR* and the second-best algorithm. Smaller improvements over the baseline were reported when evaluated against HHAR, which could be attributed to the heterogeneous set of devices used to collect the data, where the teacher model is more prone to incorrectly label samples in the knowledge distillation step.

Activity-level comparison

In order to give a more in-depth comparison between models at the activity level, the delta of confusion matrices, which are obtained by subtracting one confusion matrix from another, are provided in Figure 3.4.

Figure 3.4 shows the differences between confusion matrices obtained by using the supervised pipeline and *SelfHAR*. The confusion matrices across five independent runs were averaged before computing the difference. These delta confusion matrices were obtained by subtracting the confusion matrices of the supervised method from those obtained using our proposed pipeline. Green cells denote an improvement over the fully supervised models, while red cells indicate an increase in model confusion. Along the main diagonal, positive values indicate that the model performed better, where more samples are categorised correctly, while positive values in off-diagonal cells indicate an increase in the number of samples misclassified by the models.

Our results showed that overall, *SelfHAR* improved over supervised models in most of the activities across all datasets. Positive improvements or similar performance across all classes can be seen on the HHAR, PAAS, UniMiB SHAR and WISDM datasets, which vary in terms of ranges of activities and sensor placements. *SelfHAR* models perform

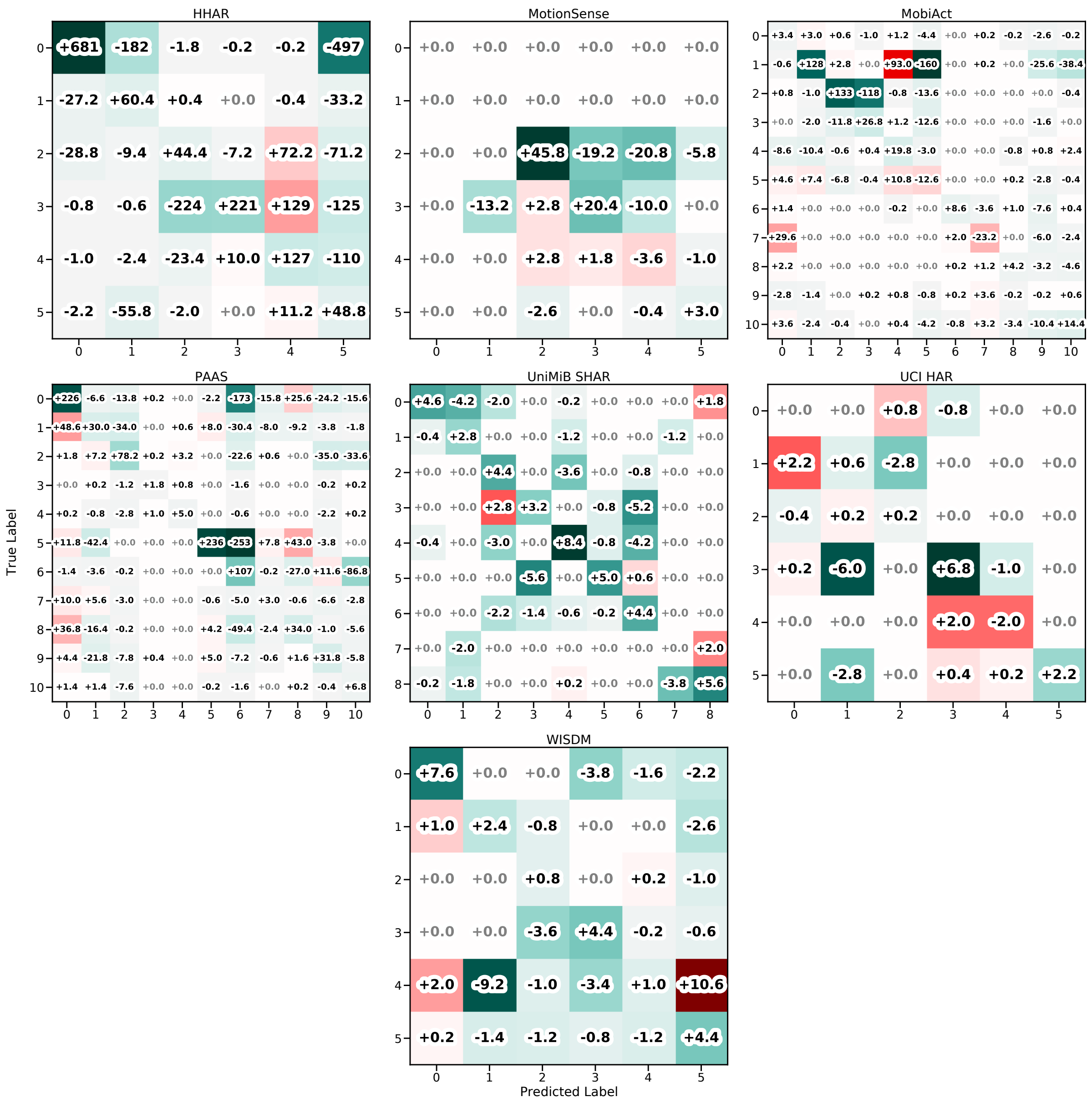


Figure 3.4: Comparison of model predictions at the activity (class) level. Delta of confusion matrices between *SelfHAR* (with MobiAct/HHAR as the unlabelled dataset) and the fully supervised models on seven different datasets are shown. Green cells denote an improvement over the fully supervised models, with an increase in correctly labelled samples or a decrease in model confusion. Red cells indicate a decrease in performance. See Table 3.3 for the activity classes which correspond to the numbers shown along the axes of the confusion matrices.

Table 3.3: Activity classes corresponding to the numbers shown along the axes of the confusion matrices in Figure 3.4.

Class Index	HHAR	MotionSense	MobiAct	PAAS	UCI HAR	UniMiB SHAR	WISDM
0	sitting	sitting	standing	sitting	walking	standing up from sitting	walking
1	standing	standing	walking	standing	walking upstairs	standing up from lying	jogging
2	walking	walking	jogging	walking	walking downstairs	walking	sitting
3	walking upstairs	walking upstairs	jumping	walking upstairs	sitting	running	standing
4	walking downstairs	walking downstairs	walking upstairs	walking downstairs	standing	walking upstairs	walking upstairs
5	biking	jogging	walking downstairs	lying down	lying down	jumping	walking downstairs
6	—	—	standing to sitting on a chair	cycling	—	walking downstairs	—
7	—	—	sitting on a chair	home activities	—	lying down to standing	—
8	—	—	sitting to standing	office activities	—	sitting down	—
9	—	—	stepping into a car	personal care	—	—	—
10	—	—	stepping out of a car	shopping	—	—	—

better in the vast majority of classes in the remaining datasets. This indicates that there is a general improvement across nearly all classes, rather than the model specialising in specific classes and sacrificing performance in other classes.

3.4.2 Ablation studies: effectiveness of the combined pre-training tasks

As our proposed pipeline can be separated into different components, it is useful to evaluate different configurations of the *SelfHAR* pipeline through ablation testing and to understand whether the novel combination of multiple supervisory signals improves the ability of the models to extract useful features from the data.

Comparisons were drawn among five different training configurations of the training pipeline (see Section 3.2.3): fully supervised, transformation discrimination, self-training, transformation knowledge distillation, and *SelfHAR*.

Table 3.4: Evaluation of the classification performance of different configurations of the proposed training pipeline when using MobiAct as the unlabelled dataset. *SelfHAR* outperforms other configurations in most datasets (metric: weighted F_1 score). *MobiAct was trained with HHAR as the unlabelled dataset.

Dataset (Evaluation Protocol)	Fully Supervised (Baseline)	Transformation Discrimination (TD)	Self-Training	Transformation Knowledge Distillation	SelfHAR
HHAR (Standard)	0.7282 [0.7241, 0.7378]	**0.7961 [0.7905, 0.8021] (+6.79%)**	0.7220 [0.7152, 0.7288] (−0.62%)	0.7724 [0.7667, 0.7786] (+4.42%)	0.7846 [0.7794, 0.7917] (+5.64%)
MotionSense (Standard)	0.9275 [0.9144, 0.9404]	0.9295 [0.9173, 0.9420] (+0.20%)	0.9208 [0.9085, 0.9347] (−0.67%)	0.9177 [0.9037, 0.9314] (−0.98%)	**0.9631 [0.9530, 0.9726] (+3.56%)**
MobiAct (Standard)	0.9098 [0.9038, 0.9140]	0.8922 [0.8852, 0.8965] (−1.76%)	0.9162 [0.9109, 0.9208] (+0.64%)	0.9086 [0.9024, 0.9134] (−0.12%)	**0.9355 [0.9305, 0.9394] (+2.57%)**
PAAS (Standard)	0.6088 [0.5985, 0.6222]	0.6851 [0.6741, 0.6967] (+7.63%)	0.6459 [0.6342, 0.6577] (+3.71%)	0.6651 [0.6537, 0.6764] (+5.63%)	**0.7022 [0.6903, 0.7125] (+9.34%)**
UniMiB SHAR (Standard)	0.7432 [0.6914, 0.7927]	0.8098 [0.7600, 0.8516] (+6.66%)	0.8264 [0.7842, 0.8666] (+8.32%)	0.8588 [0.8213, 0.8965] (+11.56%)	**0.8731 [0.8350, 0.9094] (+12.99%)**
UCI HAR (Standard)	0.9051 [0.8826, 0.9253]	0.9053 [0.8832, 0.9275] (+0.02%)	0.9079 [0.8926, 0.9342] (+0.28%)	0.9135 [0.8920, 0.9332] (+0.84%)	**0.9135 [0.8929, 0.9354] (+0.84%)**
WISDM (Standard)	0.8906 [0.8732, 0.9083]	0.8948 [0.8780, 0.9136] (+0.42%)	0.9045 [0.8950, 0.9272] (+1.39%)	0.9036 [0.8877, 0.9194] (+1.30%)	**0.9081 [0.8921, 0.9243] (+1.75%)**
HHAR (Linear)	0.7235 [0.7189, 0.7325]	**0.7507 [0.7448, 0.7584] (+2.72%)**	0.7174 [0.7106, 0.7247] (−0.61%)	0.7373 [0.7310, 0.7443] (+1.38%)	0.7327 [0.7261, 0.7403] (+0.92%)
MotionSense (Linear)	0.9224 [0.9083, 0.9355]	**0.9468 [0.9359, 0.9581] (+2.44%)**	0.9238 [0.9094, 0.9380] (+0.14%)	0.9062 [0.8905, 0.9207] (−1.62%)	0.9276 [0.9151, 0.9403] (+0.52%)
MobiAct (Linear)	0.9008 [0.8954, 0.9055]	0.8812 [0.8749, 0.8865] (−1.96%)	0.8944 [0.8885, 0.8995] (−0.64%)	0.8623 [0.8552, 0.8678] (−3.85%)	**0.9216 [0.9163, 0.9269] (+2.08%)**
PAAS (Linear)	0.6228 [0.6131, 0.6365]	0.6815 [0.6702, 0.6939] (+5.87%)	0.6158 [0.6045, 0.6289] (−0.70%)	0.6374 [0.6261, 0.6506] (+1.46%)	**0.6895 [0.6778, 0.7017] (+6.67%)**
UniMiB SHAR (Linear)	0.5806 [0.5179, 0.6437]	0.5739 [0.5148, 0.6418] (−0.67%)	0.5516 [0.4898, 0.6201] (−2.90%)	0.5445 [0.4833, 0.6117] (−3.61%)	**0.6219 [0.5624, 0.6837] (+4.13%)**
UCI HAR (Linear)	**0.8759 [0.8513, 0.8999]**	0.8577 [0.8315, 0.8820] (−1.82%)	0.8631 [0.8350, 0.8891] (−1.28%)	0.8371 [0.8108, 0.8671] (−3.88%)	0.8707 [0.8457, 0.8950] (−0.52%)
WISDM (Linear)	0.8379 [0.8148, 0.8615]	0.8672 [0.8461, 0.8871] (+2.93%)	0.8107 [0.7865, 0.8373] (−2.72%)	0.8042 [0.7798, 0.8312] (−3.37%)	**0.8811 [0.8629, 0.9017] (+4.32%)**

Table 3.5: Evaluation of the classification performance of different configurations of the proposed training pipeline when using Fenland as the unlabelled dataset. *SelfHAR* outperforms other configurations in most datasets (metric: weighted F_1 score).

Dataset (Evaluation Protocol)	Fully Supervised (Baseline)	Transformation Discrimination (TD)	Self-Training	Transformation Knowledge Distillation	SelfHAR
HHAR (Standard)	0.7282 [0.7241, 0.7378]	**0.8074 [0.8028, 0.8141] (+7.92%)**	0.7416 [0.7356, 0.7484] (+1.34%)	0.7811 [0.7747, 0.7872] (+5.29%)	0.7772 [0.7713, 0.7839] (+4.90%)
MotionSense (Standard)	0.9275 [0.9144, 0.9404]	0.9101 [0.8953, 0.9244] (−1.74%)	0.9062 [0.8924, 0.9197] (−2.13%)	0.9291 [0.9156, 0.9416] (+0.16%)	**0.9515 [0.9406, 0.9625] (+2.40%)**
MobiAct (Standard)	0.9098 [0.9038, 0.9140]	0.9112 [0.9054, 0.9152] (+0.14%)	0.9042 [0.8981, 0.9085] (−0.56%)	0.9242 [0.9192, 0.9291] (+1.44%)	**0.9249 [0.9198, 0.9290] (+1.51%)**
PAAS (Standard)	0.6088 [0.5985, 0.6222]	0.7036 [0.6922, 0.7146] (+9.48%)	0.6730 [0.6625, 0.6839] (+6.42%)	0.6636 [0.6523, 0.6755] (+5.48%)	**0.7049 [0.6931, 0.7161] (+9.61%)**
UniMiB SHAR (Standard)	0.7432 [0.6914, 0.7927]	0.8086 [0.7581, 0.8502] (+6.54%)	0.8216 [0.7770, 0.8632] (+7.84%)	0.8627 [0.8191, 0.8986] (+11.95%)	**0.8481 [0.8025, 0.8883] (+10.49%)**
UCI HAR (Standard)	0.9051 [0.8826, 0.9253]	0.9218 [0.9014, 0.9425] (+1.67%)	0.9128 [0.8914, 0.9316] (+0.77%)	0.9177 [0.8973, 0.9374] (+1.26%)	**0.9237 [0.9039, 0.9430] (+1.86%)**
WISDM (Standard)	0.8906 [0.8732, 0.9083]	0.9061 [0.8907, 0.9228] (+1.55%)	0.9104 [0.8944, 0.9270] (+1.98%)	**0.9154 [0.8996, 0.9309] (+2.48%)**	0.9090 [0.8930, 0.9253] (+1.84%)
HHAR (Linear)	0.7235 [0.7189, 0.7325]	**0.7996 [0.7942, 0.8060] (+7.61%)**	0.7004 [0.6936, 0.7071] (−2.31%)	0.7257 [0.7192, 0.7324] (+0.22%)	0.7393 [0.7334, 0.7470] (+1.58%)
Motion Sense (Linear)	0.9224 [0.9083, 0.9355]	0.9106 [0.8947, 0.9257] (−1.18%)	0.9056 [0.8906, 0.9209] (−1.68%)	0.9310 [0.9179, 0.9439] (+0.86%)	**0.9421 [0.9293, 0.9533] (+1.97%)**
MobiAct (Linear)	0.9008 [0.8954, 0.9055]	0.9061 [0.9002, 0.9108] (+0.53%)	0.8986 [0.8928, 0.9032] (−0.22%)	0.9053 [0.8997, 0.9108] (+0.45%)	**0.9167 [0.9115, 0.9216] (+1.59%)**
PAAS (Linear)	0.6228 [0.6131, 0.6365]	0.6664 [0.6551, 0.6790] (+4.36%)	0.6383 [0.6265, 0.6506] (+1.55%)	0.6636 [0.6514, 0.6757] (+4.08%)	**0.6869 [0.6750, 0.6992] (+6.41%)**
UniMiB SHAR (Linear)	0.5806 [0.5179, 0.6437]	0.5363 [0.4770, 0.6047] (−4.43%)	0.5369 [0.4802, 0.6055] (−4.37%)	0.5205 [0.4591, 0.5873] (−6.01%)	**0.6072 [0.5474, 0.6735] (+2.66%)**
UCI HAR (Linear)	**0.8759 [0.8513, 0.8999]**	0.8211 [0.7906, 0.8552] (−5.48%)	0.8569 [0.8299, 0.8853] (−1.90%)	0.8359 [0.8067, 0.8661] (−4.00%)	0.8734 [0.8464, 0.8983] (−0.25%)
WISDM (Linear)	0.8379 [0.8148, 0.8615]	0.7944 [0.7700, 0.8229] (−4.35%)	0.8210 [0.7964, 0.8478] (−1.69%)	0.7626 [0.7369, 0.7906] (−7.53%)	**0.8605 [0.8390, 0.8823] (+2.26%)**

The final models of the five training pipelines have the same computational complexity as they share exactly the same neural network architecture and the number of weights.

In Tables 3.4 and 3.5, we compare the performance of *SelfHAR* to the other four configurations. Our proposed SelfHAR pipeline achieved the highest performance in the vast majority of cases, irrespective of the unlabelled dataset used. In linear evaluation, a similar trend is observed, and in some cases, only the *SelfHAR* pipeline saw performance improvements compared to the baseline, where other configurations all performed worse (such as in MobiAct and UniMiB SHAR in Table 3.4). However, the performance gap is smaller compared to the full model evaluation. This could be due to the difference in the fine-tuning step in the *SelfHAR* pipeline: in normal training, the last convolutional layer of the student model is fine-tuned together with the classification head. However, in linear evaluation, the entire network is frozen, preventing the last convolutional layer from being fine-tuned. These results provide support for the design choice of unfreezing the last layer during fine-tuning.

This set of ablation experiments confirms our hypothesis that an increase in both internal diversity (through transformation discrimination) and external diversity (through the use of unlabelled datasets and self-training) of training data results in a better semi-supervised learning approach than focusing only on one aspect or having a fully supervised approach.

This also indicates the effectiveness of the multi-task training paradigm, where combining several approaches gives better performance than a single approach, and in some cases, more than the individual approaches added together. *SelfHAR* outperformed other configurations of the pipeline, including the previous state-of-the-art method in self-supervised HAR (the transformation discrimination task, or Transformation Prediction Network as proposed in [Saeed et al., 2019]), providing support for the specific design of the pipeline.

In addition, we observed the following trade-off in the results: while the fully supervised pipeline requires the least amount of training and performs the worst in most cases, *SelfHAR* requires the most amount of training but performs the best. The comparison is drawn between final models with the same architecture and number of weights but from different training pipelines. Due to the limited computing resources on mobile devices and the general scarcity of well-curated labelled datasets, the current need for deep learning models on wearable and mobile devices tends to focus on high accuracy and low latency. Our approach helps in this direction by improving performance without increasing the model complexity at inference.

3.4.3 Impact of training with limited labelled data

In addition to training models with full availability of labelled data, the proposed method was also evaluated in scenarios where there are limited amounts of labelled data. This

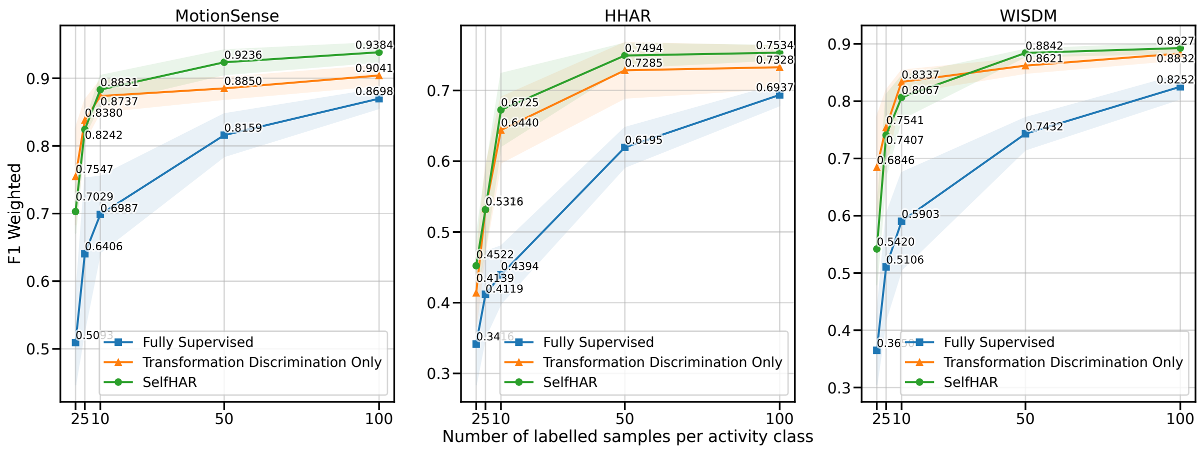


Figure 3.5: Assessing classification performance as a function of limited labelled data. *SelfHAR* achieves high performance with significantly less training data and outperforms the variant with *no* teacher-student training in most cases.

evaluation is designed to simulate scenarios where the resources for collecting labelled data are very limited, which might arise when a new set of sensors is introduced, or the set of target activities and sensor placements are changed. In this evaluation protocol, a fixed number of labelled samples per activity class were extracted from the labelled datasets, and they were the only labelled training data that the models were trained on. This procedure is used to evaluate the performance of the models when the availability of data is low.

For the fully supervised training pipeline, the models were trained on these extracted samples and evaluated directly. The transformation-discrimination-only approach follows closely to that when evaluated with full availability of labelled data (see Section 3.4.2), where the models were pre-trained on the entire training set using the task of transformation discrimination (the activity labels were not used during pre-training), and then fine-tuned on the extracted samples. The *SelfHAR* pipeline involves training a teacher model with the extracted samples, followed by the student model pre-trained with the labels generated by the teacher model and the signal transformations labels, on sensor data from both the labelled and unlabelled dataset. The student model was fine-tuned in a similar manner at the end with the extracted samples.

Either 2, 5, 10, 50, or 100 samples per activity class were extracted from the labelled dataset to simulate different degrees of availability of labelled data. Similar to other experiment protocols, five independent runs were performed for each setting.

Figure 3.5 illustrates the performance of models trained by different pipelines with limited data availability. We observe significant performance differences between the *SelfHAR* pipeline and the fully-supervised pipeline, especially when the amount of data available is particularly limited (2 - 10 samples per class). The difference in performance between the transformation discrimination only pipeline and SelfHAR varies depending on the datasets, with insignificant performance differences between them when there are only 2 - 10 samples per class, and the standard deviations are large. However, the *SelfHAR* pipeline shows a consistent performance improvement over other methods when there are 10 - 100 samples per class, with the performance gain being the largest in the MotionSense dataset.

The consistent performance gain across the range of data further indicated that the method can be adopted in general without penalty in performance. In cases of severe lack of data (labelled sample per class being less than 10), the transformation discrimination task on its own might perform better on some datasets. This could be attributed to the limitation of the teacher-student training paradigm: the uncertainty and noise in the supervisory signals could be amplified in the process when there are not many labelled samples to learn from. As the availability of data increases, the supervisory signals captured by the teacher-student training paradigm improve, and the *SelfHAR* pipeline gives a higher performance gain than when using transformation discrimination alone.

3.4.4 Visualisation of extracted features using t-SNE

In addition to the confidence intervals being reported, t-SNE visualisations [Maaten and Hinton, 2008] of the features extracted by the last convolutional layer (after max pooling) of the models are also analysed. t-Distributed Stochastic Neighbour Embedding (t-SNE) is an algorithm for projecting high-dimensional data points into two- or three-dimensional spaces, where the algorithm minimises the difference between probability distributions of the high-dimensional and the low-dimensional space, giving representations that tend to cluster close-by similar data points [Maaten and Hinton, 2008]. The minimisation is done using a gradient-based method on the Kullback-Leibler divergence of the two probability distributions. The extracted features are visualised in order to give a better understanding of the models.

Figure 3.6 displays the t-SNE projections of the extracted features by the models prior to fine-tuning, coloured by their ground-truth classes (activity labels were not used when generating the t-SNE projection). The projections show that, without direct access to the ground truth labels, *SelfHAR* discovered semantic manifolds within the data similar to the supervised model which had access to the labels. Similar results were obtained when using the rest of the datasets. We notice that the features largely correspond to those obtained with the supervised model. Notably, the clusters of data points across the two methods are similar in datasets such as WISDM, where the activity classes like jogging, walking and sitting were spread in a continuum according to their intensity. These findings suggest that the performance boost did not come from over-fitting or mode collapse. It

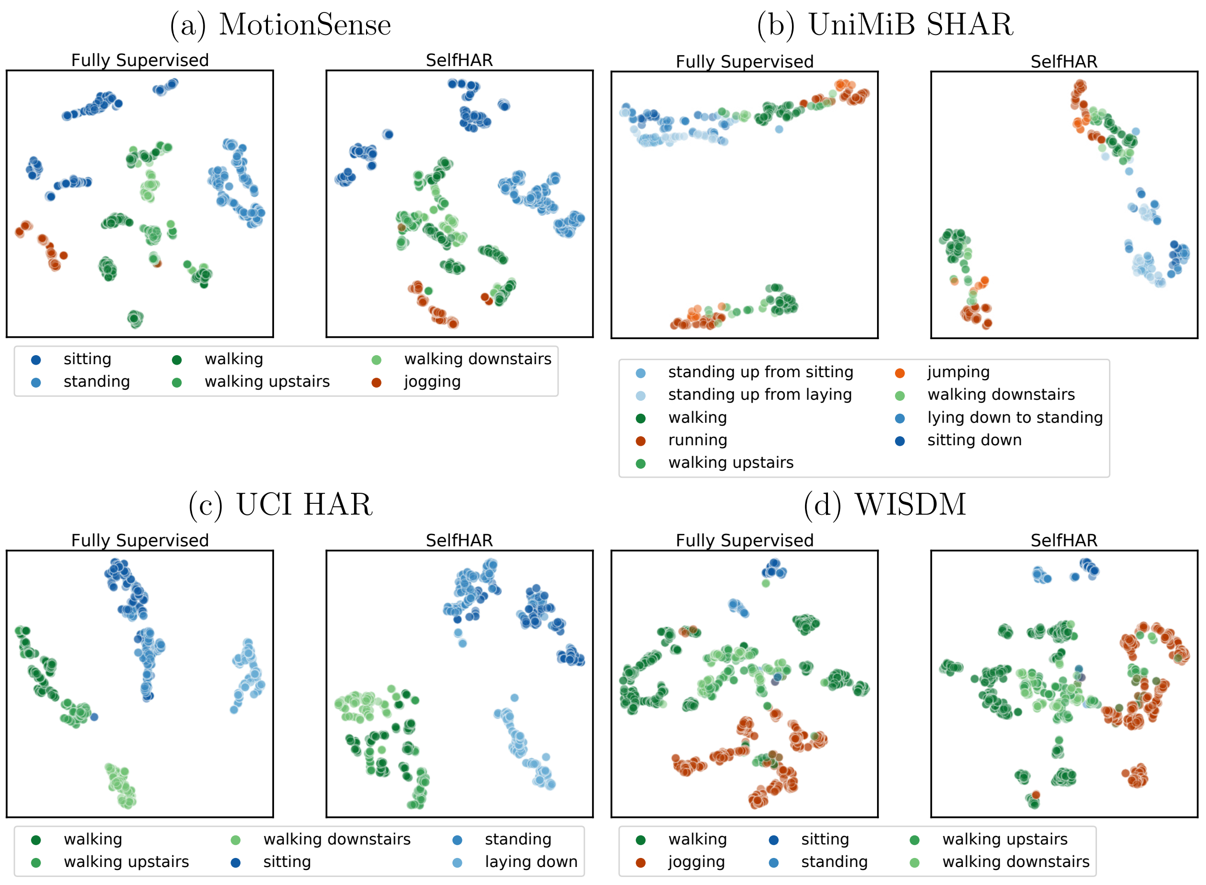


Figure 3.6: Visualising the learned representations of *SelfHAR* versus the fully supervised model. t-SNE projections of the last convolutional layer showcase that the student before fine-tuning learns similar representations to the supervised model, even without direct access to the ground-truth labels.

is evident that its inner workings resemble a supervised model.

3.4.5 Evaluating the importance of using unlabelled data

The design of the *SelfHAR* pipeline has been motivated by the use of unlabelled data, in which self-training and self-supervised pre-training are combined to leverage large-scale unlabelled datasets. However, both the novel training pipeline and the use of unlabelled datasets could have equally contributed to its performance gain over previous methods.

To verify the impact of unlabelled data, we conducted an additional set of ablation experiments on the use of unlabelled data among different training pipelines. In particular, we removed the unlabelled dataset from the *SelfHAR* pipeline, where the mixed unlabelled dataset $\mathbb{V}$ (as defined in Section 3.2.2), which is later labelled by teacher model, is formed only using the data from the labelled dataset $\mathbb{L}$, and filtering is no longer performed. Other

Table 3.6: Performance comparison of models trained with and without unlabelled data (MobiAct) across different training pipelines. *SelfHAR* with unlabelled data outperforms other configurations in most datasets (metric: weighted F_1 score). Models with the best performance among all pipelines are in bold, and those with the better performance between using and not using unlabelled data with the same training pipeline are underlined.

Training Pipeline	Fully Supervised (Baseline)	Transformation Discrimination (TD)		SelfHAR	
Data	Without unlabelled data	Without unlabelled data	With unlabelled data	Without unlabelled data	With unlabelled data
HHAR	0.7282 [0.7241, 0.7378]	0.7805 [0.7752, 0.7874] (+5.23%)	**0.7961 [0.7905, 0.8021] (+6.79%)**	0.7839 [0.7784, 0.7909] (+5.57%)	0.7846 [0.7794, 0.7917] (+5.64%)
MotionSense	0.9275 [0.9144, 0.9404]	0.9213 [0.9079, 0.9336] (−0.62%)	0.9295 [0.9173, 0.9420] (+0.20%)	0.9560 [0.9442, 0.9659] (+2.85%)	**0.9631 [0.9530, 0.9726] (+3.56%)**
MobiAct	0.9098 [0.9038, 0.9140]	0.9293 [0.9246, 0.9336] (+1.95%)	0.8922 [0.8852, 0.8965] (−1.76%)	0.9299 [0.9247, 0.9343] (+2.01%)	**0.9355 [0.9305, 0.9394] (+2.57%)**
PAAS	0.6088 [0.5985, 0.6222]	0.6384 [0.6279, 0.6513] (+2.96%)	0.6851 [0.6741, 0.6967] (+7.63%)	0.6810 [0.6699, 0.6920] (+7.22%)	**0.7022 [0.6903, 0.7125] (+9.34%)**
UniMiB SHAR	0.7432 [0.6914, 0.7927]	0.8345 [0.7912, 0.8729] (+9.13%)	0.8098 [0.7600, 0.8516] (+6.66%)	0.8575 [0.8195, 0.8936] (+11.43%)	**0.8731 [0.8350, 0.9094] (+12.99%)**
UCI HAR	0.9051 [0.8826, 0.9253]	0.9125 [0.8924, 0.9331] (+0.74%)	0.9053 [0.8832, 0.9275] (+0.02%)	0.9134 [0.8917, 0.9331] (+0.83%)	**0.9135 [0.8929, 0.9354] (+0.84%)**
WISDM	0.8906 [0.8732, 0.9083]	0.9051 [0.8886, 0.9230] (+1.45%)	0.8948 [0.8780, 0.9136] (+0.42%)	**0.9112 [0.8956, 0.9258] (+2.06%)**	0.9081 [0.8921, 0.9243] (+1.75%)

components of the training pipeline are kept unchanged (see Section 3.2.2). Similarly, we also compared them against models trained using the transformation discrimination pipeline without unlabelled data, unlike those in previous experiments.

The results in Table 3.6 show the mean and bootstrap confidence intervals of weighted F_1 scores of models trained with and without the use of unlabelled data. The comparison is drawn among the fully-supervised baseline, transformation discrimination, and *Self-HAR*. We can see that in that using *SelfHAR* with unlabelled data achieved the highest performance on most datasets. While SelfHAR without unlabelled data still improves over the fully-supervised baseline, the use of unlabelled data can further boost performance, especially for PAAS and UniMiB SHAR. On the other hand, Transformation Discrimination shows inconsistent improvements when using unlabelled data, often underperforming models with only labelled data.

This set of experiments demonstrates that *SelfHAR* is able to more consistently leverage unlabelled data compared to the previous state-of-the-art self-supervised HAR meth-

Table 3.7: Evaluation of the classification performance using different filtering techniques for free-living accelerometer data. Pre-training *SelfHAR* with balanced physical activity data outperforms the baseline and other variants.

Evaluation metrics	Unlabelled dataset for pre-training					
	None (Fully Supervised) (Baseline)	None (Transformation Discrimination)	None (SelfHAR)	Fenland - Inactive (SelfHAR)	Fenland - Balanced (SelfHAR)	Fenland - Active (SelfHAR)
F_1 Weighted	0.6088 [0.5985, 0.6222]	0.6691 [0.6580, 0.6811] (+6.03%)	0.6810 [0.6699, 0.6920] (+7.22%)	0.6832 [0.6722, 0.6948] (+7.44%)	**0.7049 [0.6931, 0.7161] (+9.61%)**	0.6662 [0.6557, 0.6786] (+5.74%)
F_1 Macro	0.4264 [0.3987, 0.4645]	0.5325 [0.4907, 0.5674] (+10.41%)	0.5298 [0.4847, 0.5669] (+10.14%)	0.5308 [0.4906, 0.5659] (+10.44%)	**0.5608 [0.5243, 0.5961] (+13.44%)**	0.5394 [0.4968, 0.5724] (+11.30%)
Cohen's Kappa	0.5182 [0.5047, 0.5323]	0.5986 [0.5854, 0.6123] (+6.31%)	0.6145 [0.6012, 0.6271] (+7.90%)	0.6156 [0.6023, 0.6280] (+9.74%)	**0.6465 [0.6327, 0.6596] (+12.83%)**	0.6004 [0.5853, 0.6128] (+8.22%)

ods. The use of unlabelled data enables further performance improvements, supporting the design of our proposal.

3.4.6 Leveraging large unlabelled datasets to improve supervised performance: PAAS and Fenland case study

Upon designing our experiments, we hypothesised that the distribution of the dataset where pre-training took part may affect the results of the downstream task. Here, our aim is to test if having a similar distribution of activities or very different distributions would affect the results of the downstream task.

For our evaluation, we constructed three different Fenland sub-datasets to test these effects using MET (metabolic equivalents) cutoffs. METs are a valuable tool for these cutoffs as they are well-established markers of energy expenditure. We decided to use Fenland and PAAS as both datasets used comparable devices placed on the same wrist. The first dataset was constructed using only low MET values ($0.5 \leq MET \leq 1.5$, conveying mostly sedentary behaviours). The second one was a balanced dataset with the same number of samples from three MET distributions: low ($0.5 \leq MET \leq 1.5$), medium ($1.5 \leq MET \leq 3.0$) and high ($MET \geq 3.0$). Finally, an active dataset was built using only MET values ≥ 3.0. All three datasets used a nearly identical number of total samples, allowing us to query the role of the activity intensity distribution.

The results of this analysis can be found in Table 3.7. We observed that self-supervised pre-training with the balanced dataset yielded the best performance, improving upon the fully supervised pipeline by 13.44% in F_1 macro. On the other hand, the low MET dataset yielded a performance improvement of 10.44% while the active dataset improvement was

11.30% on average in F_1 macro. Compared to transformation discrimination or SelfHAR without using unlabelled data, using a balanced unlabelled dataset offers noticeable performance improvements, while using other unlabelled partitions only offers moderate to little performance gain. These results show that a more balanced distribution of data gave greater improvement compared to extreme distributions of data (either very active or non-active). This indicates that an unlabelled dataset which covers the full range of activities is more desirable as a pre-training task, which confirms our hypothesis regarding how this dataset is able to increase the diversity of data better than skewed datasets. Further case studies on other large-scale unlabelled datasets might provide more insights into the influence of the distribution of unlabelled data.

3.5 Conclusions

In light of the inherent limitations present in mobile sensing datasets and their gathering, we introduced *SelfHAR*, a training pipeline that combines self-training and self-supervised learning techniques to allow deep learning models to generalise to unseen scenarios by leveraging unlabelled data, in this chapter. The combined training pipeline increases both the internal and external diversity of data that models are trained on. Compared to previous approaches, our method is able to be able to further boost the performance of activity recognition models by leveraging the abundance of unlabelled data to complement the limited labelled data. Evaluation indicates that the models trained using *SelfHAR* outperform other supervised and semi-supervised training approaches, both in terms of overall performance and individual activities, without increasing the complexity of the models at inference time.

This chapter demonstrated the potential of using different training paradigms including self-supervised learning and self-training to leverage unlabelled data, as well as the importance of diverse data for developing accurate, data-efficient HAR systems. While our proposed method focused on leveraging unlabelled data from individual devices, where accelerometer data was incorporated from one device at a time, we have not yet explored leveraging data from other sources, such as those from multiple devices. In the next chapter, we will expand our focus to the multi-device setting. We will investigate how self-supervised learning techniques can be adapted to improve the performance and data efficiency of HAR systems by leveraging the unique learning opportunity in synchronised data streams from multiple devices.

Chapter 4

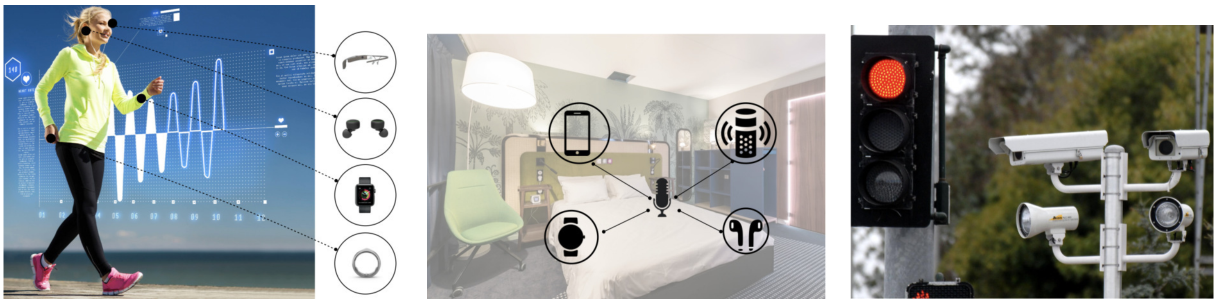
Collaborative contrastive learning for human activity recognition

In this chapter, we build upon our previous work to investigate another data source: unlabelled data collected from *multiple* devices worn by a user, as discussed in Section 2.1.2, and study how these data can be used to learn high-quality features using self-supervised learning. Motivated by the research gaps highlighted in Section 2.3, we investigate how contrastive learning, which pushes semantically similar samples closer and pulls dissimilar samples farther in the embedding space, as covered in Section 2.2.3, can be leveraged in this collaborative, multi-device setting.

4.1 Motivation and overview

Contrastive learning, as covered in Section 2.2.3, is a self-supervised learning method that conventionally involves comparing a data sample against its *transformed version* and other samples in the dataset. It has become prominent for its effectiveness in training a strong feature extractor using unlabelled data [Chen et al., 2020a, Grill et al., 2020, Tang et al., 2020]. However, works in this area have shown that the choice of transformation functions has a significant impact on the performance of the model. Hence, it requires a careful design to ensure a successful implementation, and this requires manual feature engineering and human input.

In this chapter, we extend this line of research, by studying a problem setting called Time-Synchronous Multi-Device System (TSMDS), which is a collaborative sensing setting (as discussed in Section 2.1.2) and exhibits an unexplored opportunity for self-supervised learning. This problem setting is inspired by the current trend of people wearing multiple inertial measurement unit (IMU)-enabled devices, including smartphones and consumer wearables. Studies including the one by [Safaei et al., 2017] estimate that by the year 2025, each person will own 9.3 connected devices on average. An example of



(a) On-body inertial sensors (b) Acoustic sensors in a living space (c) Cameras at a traffic junction

Figure 4.1: Examples of Time-Synchronous Multi-Device Systems (TSMDS) in different contexts.

this trend is shown in Figure 4.1a – here, a user is wearing multiple IMU-enabled devices which are collecting time-synchronised sensor data while the user is performing a physical activity, such as jogging.

Apart from the growing importance and practicality of this problem setting, it presents a unique opportunity for contrastive learning for HAR. In the TSMDS setting, multiple devices on a user's body are capturing the same physical phenomenon (i.e., a user's activity) from different viewpoints. For example, when a user is running, both the smartphone placed in one of the trousers' pockets and the smartwatch worn on the wrist record sensor data for the running activity, but from different perspectives due to their unique placements on the body. Thus, rather than manually generating transformations of the data samples for contrastive learning, we can interpret the data from different devices in the TSMDS setting as natural transformations of each other, and leverage it to design self-supervised contrastive learning algorithms. Here, different devices collaborate in the process of self-supervised learning, and hence we call this approach Collaborative Self-Supervised Learning (ColloSSL).

This chapter starts with a formal definition of the TSMDS setting in Section 4.2.2. We then elaborate on the limitations of current contrastive learning techniques in Section 4.2.3, which serve as the motivation for our research problem. In Section 4.3, we introduce novel algorithms and optimisation objectives for Collaborative Self-Supervised Learning, namely Device Selection, Contrastive Sampling, and a Multi-view Contrastive Loss function. Later, we present an end-to-end semi-supervised framework which uses Collaborative Self-supervised Learning (ColloSSL) on unlabelled sensor data from multiple devices to learn high-quality features from the data, which is followed by training a HAR classifier on a small amount of labelled data to recognise specific human activities. Finally, in Section 4.5, we compare the performance of our framework against a number of supervised and semi-supervised training baselines on three multi-device HAR datasets. Our results show that ColloSSL generally outperforms supervised training and other semi-supervised baselines in both low-data and high-data regimes. We also present insights on the feature embeddings generated by ColloSSL as well as the robustness and generalisability of ColloSSL.

The work in this chapter makes three major contributions:

- We present Collaborative Self-Supervised Learning (ColloSSL), a novel method for learning from unlabelled inertial sensor data collected from multiple devices worn by the user. Different from prior methods [Saeed et al., 2019, Tang et al., 2020], ColloSSL leverages natural transformations in the sensor datasets collected from multiple devices to perform contrastive learning, and learns a robust feature extractor for downstream HAR classification tasks.
- We introduce three key research challenges for ColloSSL and propose novel device selection and data sampling algorithms, along with a new loss function which extends contrastive learning to multi-device settings.
- We present a thorough evaluation of ColloSSL on three multi-device HAR datasets covering both locomotion activities and complex activities of daily living. Our results show that ColloSSL outperforms both fully-supervised and semi-supervised baselines in terms of recognition accuracy and labelled data efficiency.

4.2 Time-synchronous multi-device systems

This section begins by providing more details on the problem setting explored in this work called Time-Synchronous Multi-Device Systems (TSMDS), followed by a mathematical formalisation, and the related research challenges. It is important to note that our objective is not to claim that it is a problem setting that has not been studied before. Instead, we argue that this is an interesting problem space in which self-supervised learning has not been studied.

4.2.1 Concepts

In the context of human activity recognition, the TSMDS problem setting is similar to a Body-Area Network in which multiple computing and sensor devices are worn on, affixed to or implanted in a person's body [Stisen et al., 2015]. The essential characteristic of TSMDS is that all devices observe a physical phenomenon (e.g., a user's locomotion activity) *simultaneously* and record sensor data in a *time-aligned manner* (see Figure 4.1a). Even though our work focuses on HAR using motion data, the TSMDS setting is rather generic and can be found in many other sensory applications. In a smart home (see Figure 4.1b), multiple voice assistants can listen to a user's speech and audio commands simultaneously. In a camera network deployed at a traffic junction (see Figure 4.1c), multiple cameras capture the same scene from different perspectives simultaneously.

Below we state the two key assumptions we have made for exploring the TSMDS setting in the context of HAR:

- We assume that all sensor devices in the multi-device system share the same sensor sampling rate, or that their data can be re-sampled to the same rate. This assumption ensures that the dimensions of the data that will be fed to the HAR model remain consistent across different devices, and it simplifies the design of the neural network architecture of the HAR model.

- By definition, we assume that multiple devices in the TSMDS setting are collecting sensor data in a time-aligned manner. Admittedly, the assumption that the sensor datasets across multiple devices are perfectly time-aligned is strong in real-world applications. There could be timestamp noise or system clock misalignment across devices, which could skew the temporal alignment of multi-device datasets. However, prior research in HAR [Stisen et al., 2015] has shown timestamp noise for accelerometer and gyroscope sensors is very small, usually less than 10ms. We hypothesise that such a small noise will not degrade the performance of our solution, and empirically validate this hypothesis by showing that our approach is robust against moderate amounts of temporal misalignment between devices in Section 4.5.9.

4.2.2 Problem statement

Building upon the concepts given above, we formalise the TSMDS problem setting as follows: we are given a set of devices $\mathbb{D}$ with time-aligned and unlabelled sensor datasets $\{\mathbb{X}_i\}_{i=1}^{|\mathbb{D}|}$. Without loss of generality, we assume that the datasets are pre-segmented into W windows, as is the convention in HAR tasks. Each dataset $\mathbb{X}_i$ contains W windows $\{x_i^1, \ldots, x_i^W\}$, where x_i^j denotes a set of sensor samples from device i in window j. In general, a sensor sample could be any form of IMU measurement, e.g., 3-dimensional accelerometer data, 3-dimensional gyroscope data, or 1-dimensional accelerometer norm. In our work, each sensor sample is a 6-dimensional vector, created by stacking together 3 axes of accelerometer and 3 axes of gyroscope data.

Let $D^* \in \mathbb{D}$ be an anchor device (e.g., a smartphone) in the contrastive learning setup for which we want to train a HAR prediction model. Let $\mathbb{L}^* = \{(x_*^1, y_*^1) \cdots, (x_*^m, y_*^m)\}$ be a (small) pre-segmented labelled dataset from the anchor device with m windows ($m \ll T$) where x_*^j is the set of sensor samples in window j and y_*^j is the class label assigned to those samples.

Our objectives are two-fold: first, we aim to use the unlabelled datasets $\{\mathbb{X}_i\}_{i=1}^{|\mathbb{D}|}$ from all the devices to train a feature extractor $f(.)$ using ColloSSL. The trained feature extractor should be able to generate high-quality feature embeddings for the anchor device data. Second, we aim to use this pre-trained feature extractor $f(.)$ to obtain feature embeddings for the labelled anchor samples x_*^j, and subsequently train a HAR classifier $g(.)$ which maps these features embeddings to the corresponding class labels y_*^j.

4.2.3 Research challenges

The TSMDS setting described above presents a unique scenario where we have natural transformations of HAR data across different devices. In this work, we investigate whether it is possible to leverage *natural transformations* that are already available in sensor datasets, instead of specifying *manual transformations* during contrastive learning. Interestingly. As shown in Figure 4.1a, multiple devices worn by the user are simultaneously capturing the *same physical activity* (e.g., running) from different views. As such, we can consider the datasets from different devices as natural transformations of each other. This observation can also be validated from an early seminal HAR work by [Kunze and Lukowicz, 2008], where they argued that the accelerometer and gyroscope data collected by body-worn sensors depend on the translation and rotation characteristics of the body part where the sensor is placed. Since different body parts have different translation and rotation characteristics (e.g., the wrist has a higher rotational degree of freedom than the chest), it induces natural transformations in the accelerometer and gyroscope data collected from different body positions.

However, there exist unique research challenges in this setting. Here, we present an example illustration in Figure 4.2. The figure shows a 15-second trace of unlabelled accelerometer data collected simultaneously from N (=4) body-worn devices. Each accelerometer trace is segmented using a window length of 3 seconds, thereby giving us 5 windows for each device. Let us say we would like to train a feature extractor for the 'chest' body position using contrastive learning. As such, the data samples from 'chest' would become the *anchor samples*. The first anchor sample from 'chest' is highlighted by a grey rectangle in Figure 4.2. As explained above, to perform contrastive learning, we need to select appropriate positive and negative samples. Below we identify three key research questions in this direction:

- From the remaining $N-1$ devices (i.e., upperarm, forearm, head), how do we select positive and negative samples for contrastive training? Intuitively, there will be some devices whose data distribution will be too far away from the data distribution of the 'chest' device. If we use these far-apart devices to obtain *positive samples* and push them closer in the embedding space to the anchor samples, it may lead to degenerate solutions for contrastive learning. Further, as the data distribution of each device

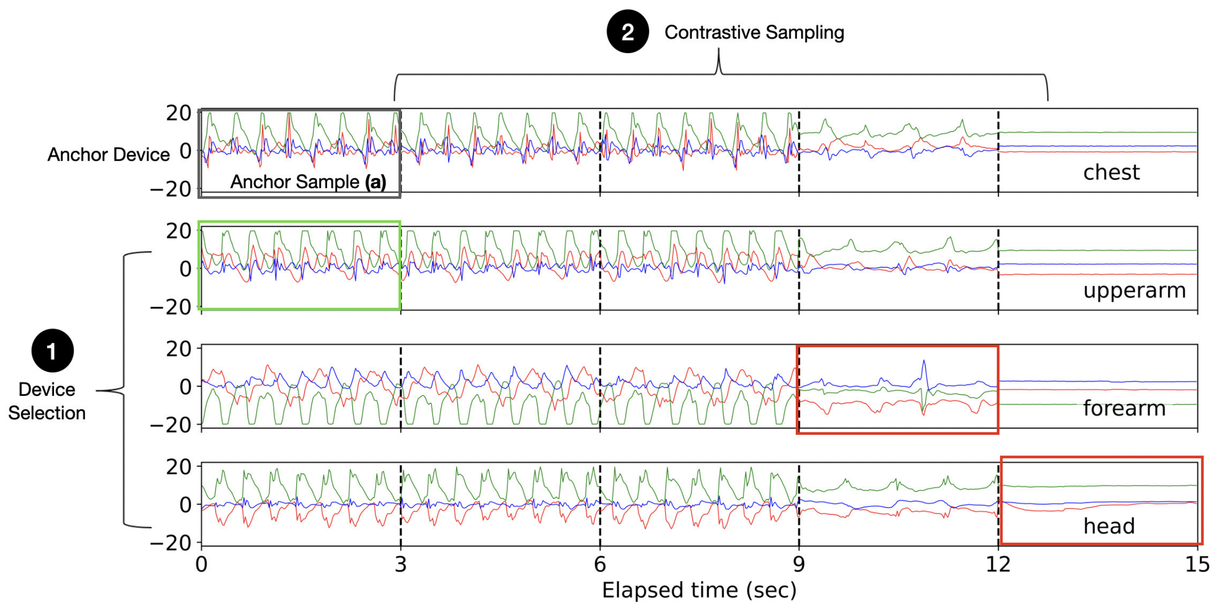


Figure 4.2: Illustration of two research challenges in ColloSSL: Device Selection and Contrastive Sampling. The anchor sample is highlighted by the grey rectangle. The green and red rectangles mark the positive and negative samples that are selected by our Device Selection and Contrastive Sampling algorithms described in Section 4.3.3.

changes depending on the user's activity, we need to account for these dynamic changes while selecting devices. We call this research challenge *Device Selection* (see Figure 4.2) and propose a data-driven strategy which uses the Maximum Mean Discrepancy (MMD) as a metric to dynamically select positive and negative devices during contrastive learning. As an illustration, we show in Figure 4.2 that our strategy chooses 'upperarm' as the positive device and 'forearm' and 'head' as the negative devices for the given data samples. The selection algorithm is described in detail in Section 4.3.3.

- In addition to *Device Selection*, we need to decide which samples from the selected devices will act as positive or negative samples. For example, if we select 'upperarm' as the positive device, which one of its 5 samples (or windows) will act as the positive sample for the anchor sample? We call this challenge *Contrastive Sampling* and propose the idea of using time-synchronised samples from positive devices and time-asynchronised samples from negative devices. For example, in Figure 4.2, the time-synchronised sample from the positive device is highlighted with a green rectangle, whereas the time-asynchronised samples from negative devices are highlighted with red rectangles. The rationale and details behind the contrastive sampling algorithm are provided in Section 4.3.3.

- Finally, to enable contrastive learning in a group of devices, we need to define a new optimisation objective. To this end, we propose a novel loss function called *Multi-View Contrastive Loss* which can take an arbitrary number of positive and negative samples as input and compute a loss metric, which is then optimised using stochastic gradient descent. Details are provided in Section 4.3.4.

4.3 Approach

In this section, we introduce our proposed approach of Collaborative Self-Supervised Learning (ColloSSL) for HAR.

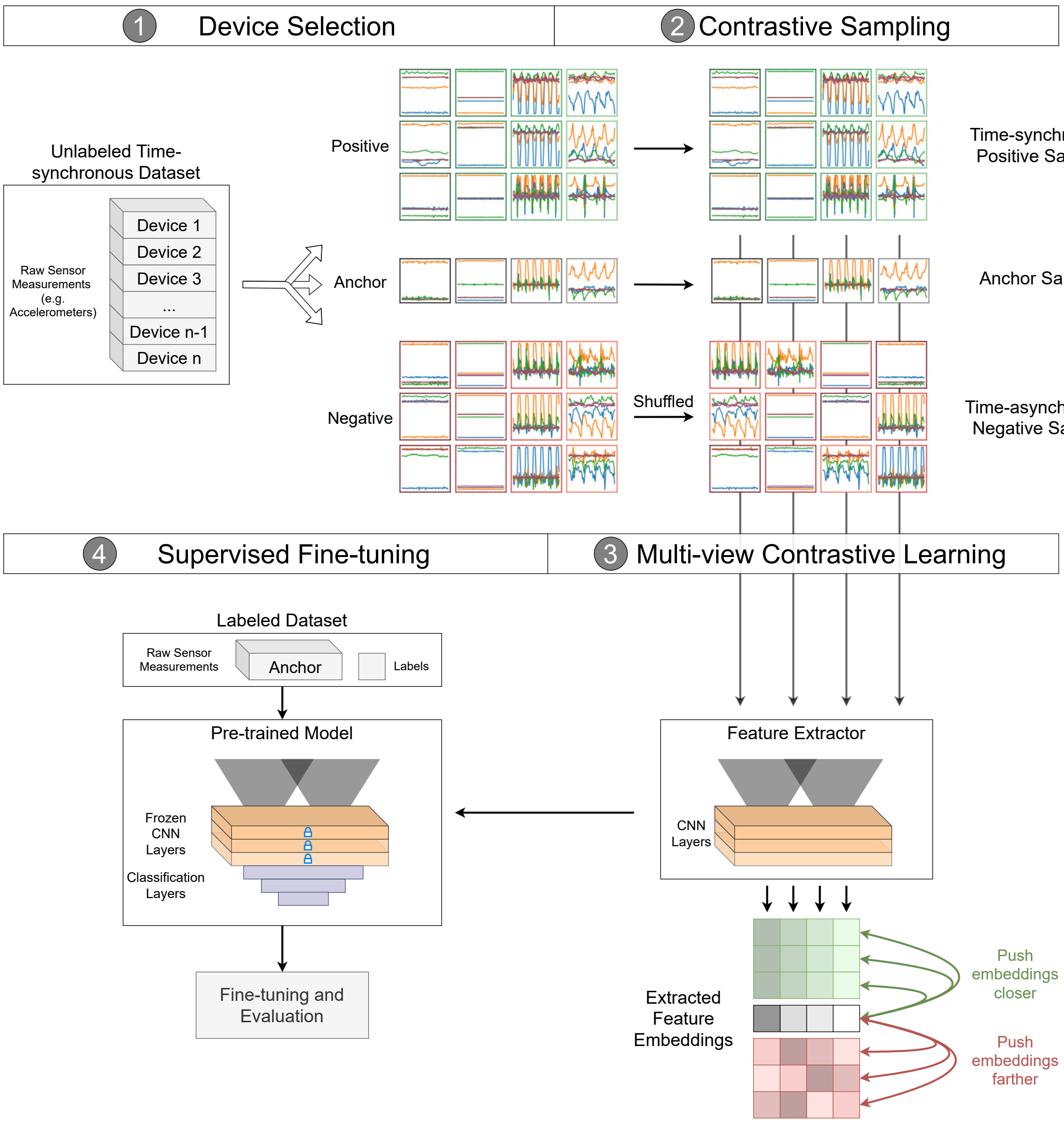


Figure 4.3: Overview of collaborative self-supervised learning (ColloSSL). A detailed explanation is provided in Section 4.3.1.

4.3.1 Solution overview

Our proposed solution is illustrated in Figure 4.3 and works as follows:

1. We initialise the feature extractor $f(.)$ with random weights.
2. We sample a batch $\mathbb{B}$ of time-aligned unlabelled data $\{x_i^1, \cdots, x_i^{|\mathbb{B}|}\}_{i=1}^{|\mathbb{D}|}$ from all devices in $\mathbb{D}$.
3. Given an anchor device D^*, we first decide which of the remaining devices will provide *positive samples* and *negative samples* for contrastive learning in this batch of data. This is done through the *Device Selection* algorithm explained in Section 4.3.3.
4. For each anchor sample, we then decide out of all samples in the batch $\mathbb{B}$, which specific samples from the positive and negative devices should be used for contrastive learning. While data sampling has been studied as an important aspect of SSL in the past, we present an algorithm called *Contrastive Sampling* (described in Section 4.3.3) which extends data sampling to the TSMDS setting.
5. The anchor, positive and negative samples are fed to the feature extractor $f(.)$ to generate feature embeddings. These feature embeddings are used to compute a Multi-view Contrastive Loss (detailed in Section 4.3.4), which pushes the positive embeddings closer to the anchor, and negative embeddings farther from the anchor, performing multi-view contrastive learning.
6. Steps 4-5 are repeated until all anchor samples in the batch $\mathbb{B}$ are computed, and Steps 2-5 are repeated until the Multi-view Contrastive Loss converges. Upon convergence, we expect that $f(.)$ has learned to extract high-quality features for the anchor device.
7. Finally, we use the pre-trained feature extractor $f(.)$ and the labelled dataset from the anchor device ($\mathbb{L}^*$) to train a HAR classifier using supervised fine-tuning (described in Section 4.3.4).

4.3.2 Feature extractor

The same feature extractor architecture as described in the previous chapter (see Section 3.3.3) has been adopted in this work. We used the same lightweight TPN model [Saeed et al., 2019] with three temporal (1-dimensional) convolutional layers followed by global pooling. Similarly, the input to the feature extractor $f(.)$ was a 2-dimensional tensor, with time on one axis and sensor data on the other. However, different from the previous work, we used both 3-axis accelerometer and 3-axis gyroscope traces as the input, and therefore the dimension of sensor data was 6. As such, given a sampling rate of 50 Hz and a window length of 2 seconds, the input tensor to the model was of dimension 100×6. The output of $f(.)$ was a feature embedding with dimension (96×1).

4.3.3 Device selection and contrastive sampling

As explained in Section 4.2.3 and Figure 4.2, two key challenges in ColloSSL are: (1) *Device Selection*, i.e., selecting the devices from which positive and negative samples will be taken, and (2) *Contrastive Sampling*: deciding which samples from the selected devices will be used for contrastive learning.

Before we present our approach, we first explain that for a given anchor sample (x), what constitutes a 'good' positive (x^+) and negative (x^-) sample for contrastive learning. Recall that the objective of contrastive learning is to guide the feature extractor f to map x and x^+ closer to each other in the feature space, and x and x^- farther from each other.

Quality of positive samples

Here, we propose that a good positive sample x^+ should have the following two characteristics:

(P1) x^+ should belong to the same label (activity class) as the anchor sample x. Since if x and x^+ are from different classes and yet the feature extractor f tries to map them closer to each other, it would lead to poor class separation and degrade the downstream classification performance.

(P2) x^+ should come from a device whose data distribution has a small divergence from that of the anchor device. This is important because if the anchor device and positive device are very different in their data characteristics (e.g., wrist-worn IMU vs. chest-worn IMU), then it might affect the feature extractor in extracting meaningful invariant embeddings.

Note that in ColloSSL, we do not have access to the ground-truth labels of the data. As such, enforcing **(P1)** on x^+ may seem tricky at first. However, from the definition of the TSMDS setting, we know that all devices collect the HAR data simultaneously and in a time-aligned fashion. Hence, we can naturally assume that the ground-truth labels (e.g., walking, running) are also time-aligned across devices. Therefore, if we can ensure that x and x^+ are time-aligned, it will implicitly guarantee that they have the same labels.

Quality of negative samples

We propose that a good negative sample x^- should have the following two characteristics:

(N1) x^- should be a true negative, i.e., belong to a different class than the anchor sample x. Because if x and x^- are from the same class and yet the feature extractor f tries to push them away, it could lead to poor classification performance.

(N2) The most informative negative samples are those whose embeddings are initially near to the anchor embeddings, and f needs to push them far apart. In this scenario, f gets a strong supervisory signal from the data and more useful gradients during training. In the alternate scenario when negative embeddings are already far apart from the anchor sample when the training starts, f will receive a weaker supervisory signal from the data, which could adversely impact its convergence.

Again, due to the unavailability of class labels in ColloSSL, strictly enforcing **(N1)** is not possible. A solution is needed which can encourage this characteristic on negative samples, and minimise the possibility of x and x^- belonging to the same class.

Having defined what constitutes a good positive and negative sample, we now describe our device selection and contrastive sampling algorithms.

Device selection

Our device selection algorithm is designed to increase the likelihood of selecting 'good' positive and negative samples. For brevity, we refer to the devices from which positive (or negative) samples are taken as *positive devices* (or *negative devices*). Formally, we are given a set of devices $\mathbb{D}$ with time-aligned and unlabelled sensor datasets $\{\mathbb{X}_i\}_{i=1}^{|\mathbb{D}|}$. Let $D^* \in \mathbb{D}$ be the anchor device. Let $\mathbb{D}^\theta = \mathbb{D} \setminus D^*$ be a *candidate set* of remaining devices from which we want to choose positive and negative devices.

Our device selection algorithm works as follows: first, we sample a batch of time-aligned data from the anchor device D^* and each of the devices in $\mathbb{D}^\theta$. Let x^* and $\mathbb{X}^\theta = \{x_i\}_{i=1}^{|\mathbb{D}^\theta|}$ be the data batches from the anchor and the candidate devices, respectively.

We compute the pairwise Maximum Mean Discrepancy (MMD) between x^* and each of the data batches in $\mathbb{X}^\theta$. MMD is a distribution-level statistic to compute the distance between two data distributions; a higher MMD implies a larger distance between distributions [Gretton et al., 2006]. After obtaining the batch-wise MMD scores between each pair of device batches, we use the device selection policies introduced below.

Closest positive

The device whose data has the least MMD distance from the anchor data is chosen as the positive device. This choice satisfies the criteria **(P2)** for selecting good positive samples and ensures that $f(.)$ can reasonably map the embeddings of the two samples closer to each other. Note that we also experimented with using more than one positive device, but found that using just one (the closest) device as positive gives the best performance.

Weighted negatives

For negative devices, we use 'all' devices from the candidate set $\mathbb{D}^\theta$, but their contributions during training are weighted by the inverse of their MMD distance from the anchor samples. Devices which have smaller MMD distances to anchor get higher weights during contrastive training, and devices with higher MMD distances get smaller weights. This policy serves two objectives: firstly, by assigning higher weights to devices with smaller MMD distances to the anchor, it satisfies **(N2)** and ensures that those negative samples which are closer to the anchor get more weight during training. Secondly, the use of 'all' devices as negatives serves as a hedge against the scenario when **(N1)** is violated on one device. For example, even if one device violates **(N1)** and ends up choosing x^- from the same class as x, the other devices can cover for it, and ensure that its impact on the feature extractor is minimal.

The weights assigned to each negative device $i \in \mathbb{D}^\theta$ can be expressed as:

$$\lambda_i = \frac{1}{\mathrm{MMD}\left(x^*, x_i\right)} \tag{4.1}$$

The weights are further normalised by dividing each weight by the maximum weight across devices.

As an example, we applied our device selection policy to the RealWorld HAR dataset [Sztyler and Stuckenschmidt, 2016] (details of the dataset are provided in Section 4.4). The dataset contains sensor data from 7 IMU-equipped devices: $\mathbb{D} = \{$chest, upperarm, forearm, thigh, shin, head, waist$\}$. We chose 'chest' as the anchor device and obtained the pairwise MMDs between data from 'chest' and data from the remaining devices. This resulted in the following pairwise MMD scores: $\{$*chest-head: 0.45, chest-waist: 0.61, chest-thigh: 0.67, chest-upperarm: 0.77, chest-forearm: 0.83, chest-shin: 1.51*$\}$. In line with our selection algorithm, we chose *head* as the positive device and $\{$*head, waist, thigh, upperarm, forearm, shin*$\}$ as the negative devices with weights inversely proportional to their MMD scores.

Contrastive sampling

In the previous step, we have decided which devices in $\mathbb{D}^\theta$ will act as positives and negatives. Now, we decide which data samples should be picked from each device for contrastive training.

Formally, we are given an anchor device D^*, a set of positive ($\mathbb{D}^+$) and negative devices ($\mathbb{D}^-$). Let $\mathbb{P}_i = \{p_i^1, \cdots, p_i^T\}|_{i=1}^{|\mathbb{D}^+|}$, $\mathbb{N}_j = \{n_j^1, \cdots, n_j^T\}|_{j=1}^{|\mathbb{D}^-|}$, $\mathbb{S} = \{s^1, \cdots, s^T\}$ be the time-aligned data batches from the i^{th} positive, j^{th} negative, and the anchor device respectively. Here, p_i^t, n_j^t and s^t each denote a data sample with timestamp t.

The objective of contrastive sampling is to select 'good' positive and negative embeddings for a given anchor sample s^t. Our proposed method uses the following sampling policies.

Synchronous positive samples. For a given anchor sample s^t at time-step t, we choose its time-aligned positive counterparts p_i^t as the positive samples. As explained earlier, this choice ensures that the anchor and positive samples have the same labels, and satisfies the **(P1)** criteria for good positive samples.

Asynchronous negative samples. A criterion for good negative samples **(N1)** is that they should belong to a different class from the anchor sample. As we do not have access to ground-truth labels during contrastive learning, it is impossible to strictly enforce **(N1)**. As a solution, we make use of the observation that negative samples which are *not time-synchronised* with the anchor are more likely to be from a different class. That is, for an anchor sample s^t at time-step t, a good negative sample would be $n^{t'} \mid t' \neq t$.

This choice however still does not guarantee that the labels at time-steps t and t' will be different; for example, a user's activity at $t = 0$ and $t' = 100$ may happen to be the same by random chance. To minimise the possibility of such cases, we use a simple trick: a large batch size of 512 is used during ColloSSL which ensures that each batch has diverse class labels in it and the possibility of a label overlap at t and t' by random chance is reduced.

Summary

The techniques presented in this section address two core research challenges of ColloSSL identified in Figure 4.2. Using the Device Selection algorithm, we first decide which of the devices will act as positive or negative during training. Next, the Contrastive Sampling algorithm finds the 'good' positive and negative samples from the selected devices, which can be used for contrastive learning with the anchor embeddings.

It may be interesting to note that the positive and negative devices selected by our algorithm are not mutually exclusive. The positive device which has the least MMD distance from the anchor will also get selected in the set of negative devices. During Contrastive Sampling, however, the samples selected from this device are different: when it acts as the positive device, samples which are time-synchronised with the anchor will be selected. However, when it acts as a negative device, samples which are not time-synchronised with the anchor will be selected.

4.3.4 Multi-view contrastive loss

In this section, we explain how the positive and negative samples selected from the previous step are used to train the feature extractor. Firstly, the anchor sample, positive samples and negative samples are fed to the feature extractor to obtain feature embeddings. Let $\{z_i^+\}|_{i=1}^{|\mathbb{D}^+|}$ and $\{z_j^-\}|_{j=1}^{|\mathbb{D}^-|}$ be the selected feature embeddings from the i^{th} positive and j^{th} negative device. Let z^* be the anchor embedding.

We propose a novel loss function called Multi-view Contrastive Loss, which is inspired by the standard contrastive loss function but compatible with multiple positive and negative samples.

$$\mathcal{L}_{\text{MCL}} = -\log \frac{\sum_{i=0}^{|\mathbb{D}^{+}|} \exp\left(\text{sim}\left(z^{*}, z_i^{+}\right)/\tau\right)}{\left(\begin{array}{c}\sum_{i=0}^{|\mathbb{D}^{+}|} \exp\left(\text{sim}\left(z^{*}, z_i^{+}\right)/\tau\right) \\ + \sum_{j=0}^{|\mathbb{D}^{-}|} \lambda_j \exp\left(\text{sim}\left(z^{*}, z_j^{-}\right)/\tau\right)\end{array}\right)} \tag{4.2}$$

where sim(.) denotes *cosine similarity*, λ_j is the weight assigned to the j^{th} negative device according to Equation 4.1, and τ is a hyperparameter denoting temperature for contrastive learning.

The loss $\mathcal{L}_{\text{MCL}}$ is minimised for each batch of data using stochastic gradient descent. Effectively, the loss minimisation during training guides the feature extractor $f(.)$ to increase the cosine similarity between anchor and positive embeddings (i.e., push them closer in the feature space), and do the opposite for anchor and negative embeddings. In doing so, $f(.)$ understands the structure of the sensor data from different devices, and learns to map raw data into high-quality features, which can be useful for various downstream classification tasks.

Supervised fine-tuning

Finally, after the feature extractor is trained using ColloSSL, it can be used for training downstream HAR classification models. Similar to our previous work, we follow the approach by [Saeed et al., 2019] of freezing the weights of the feature extractor except for its last convolution layer and adding a classification head to the model. The classification head consists of a fully connected layer of 1024 hidden units with ReLU activation, followed by an output layer with the number of units equal to the number of activity classes. The model is then trained with a small labelled dataset $\mathbb{L}^{*}$ from the anchor device by optimising the standard categorical cross-entropy loss.

4.4 Experimental protocols

In this section, we describe the multi-device HAR datasets used in this work, the baseline algorithms that we compared against, as well as other experimental protocols.

4.4.1 Datasets

For our experiments, we used three datasets for human activity recognition (HAR) that have time-aligned sensor data from multiple devices: RealWorld [Sztyler and Stuckenschmidt, 2016], Opportunity [Roggen et al., 2010], and PAMAP2 [Reiss and Stricker,

Table 4.1: Summary of the datasets used for evaluation. The RealWorld and Opportunity datasets also contain heterogeneous sensors from different manufacturers.

Dataset	Devices (positions)		Users	Activities (labels)	
RealWorld [Sztyler and Stuckenschmidt, 2016]	7	(forearm, thigh, head, upper arm, waist, chest, shin)	15	8	(stair up, stair down, jumping, lying, standing, sitting, running, walking)
Opportunity [Roggen et al., 2010]	5	(back, left lower arm, right shoe, right upper arm, left shoe)	4	4	(standing, walking, sitting, lying)
PAMAP2-Locomotion [Reiss and Stricker, 2012]	3	(arm, chest, ankle)	8	4	(standing, walking, sitting, lying)
PAMAP2-ADL [Reiss and Stricker, 2012]	3	(arm, chest, ankle)	8	12	(standing, walking, sitting, lying, running, cycling, nordic walking, ascending stairs, descending stairs, vacuum cleaning, ironing, rope jumping)

2012] as shown in Table 4.1. In common, they contained 3-axis accelerometer and 3-axis gyroscope data sampled simultaneously from multiple on-body devices. The inertial sensors used in two of the datasets were also heterogeneous: the RealWorld dataset used a Samsung Galaxy S4 smartphone and an LG G smartwatch to collect the sensor data, while the inertial sensors used in the Opportunity dataset also came from different manufacturers such as InertiaCube3 and Sun SPOT.

RealWorld

The RealWorld dataset [Sztyler and Stuckenschmidt, 2016] contains accelerometer and gyroscope traces from 15 participants, sampled at 50 Hz simultaneously on 7 sensor devices mounted at the forearm, thigh, head, upper arm, waist, chest, and shin. Each participant performed 8 activities: climbing stairs down and up, jumping, lying, standing, sitting, running/jogging, and walking.

Opportunity

The Opportunity dataset [Roggen et al., 2010] consists of IMU data collected from 4 participants performing activities of daily living with 17 on-body sensor devices, sampled at 30 Hz. For our evaluation, we selected five devices deployed on the back, left lower arm, right shoe, right upper arm, and left shoe, and we aimed to detect the mode of locomotion, namely standing, walking, sitting, and lying.

PAMAP2 (Locomotion and ADL)

The PAMAP2 Physical Activity Monitoring dataset [Reiss and Stricker, 2012] contains data on 18 different physical activities, performed by 9 participants with 3 IMUs. The IMUs were deployed over the wrist on the dominant arm, on the chest, and on the dominant side's ankle with a sampling rate of 100 Hz. Out of 9 participants, we used the data from 8 of them, since the remaining user had data for only one activity class. We performed the evaluations using two different splits of the dataset: PAMAP2-Locomotion, which consisted of 4 locomotion activities: standing, walking, sitting, and lying, and PAMPA2-ADL, which consisted of 12 ADLs (activities of daily living): running, cycling, nordic walking, ascending stairs, descending stairs, vacuum cleaning, ironing, and rope jumping, along with the four locomotion activities.

Some of these datasets, Opportunity and PAMAP2-Locomotion, have a low number of activities to be recognised, which constitute relatively simple activity recognition tasks. Previous works [Sagha et al., 2011, Bhattacharya et al., 2014, Kwon et al., 2018, Haresamudram et al., 2019, Bennasar et al., 2022] have shown that they can be well handled by traditional feature-engineering approaches (as introduced in Section 2.1.3). However, since this work requires curated, time-aligned sensor data from multiple devices, which are relatively less available, we have decided to evaluate our proposal and compare it against other baselines on these datasets. This also leads to a different selection of datasets from Chapter 3. Furthermore, deep learning methods generally provide a more flexible and extensible framework for activity recognition, in which it is possible to pre-train models on unlabelled datasets and fine-tune them for specific tasks. As a result, our evaluation focuses on deep learning methods. For simpler tasks, it is worth noting that a traditional feature-engineering pipeline with statistical features may provide strong results with less overhead.

4.4.2 Baselines

We compared ColloSSL against 6 baselines, divided into 4 categories as described below.

Random

In the Random baseline, we assigned random weights to the feature extractor and froze them. During the supervised fine-tuning, only the classification head was trained using labelled data from the anchor device. This baseline is used to confirm that our learning task is not so trivial that it can be solved with a random feature extractor.

Fully-supervised learning

Similar to our previous work, we compared our proposal against the fully-supervised training setup (see Section 3.4.1). Here, we devised two baselines, Supervised-single and Supervised-multi. In both baselines, the feature extractor and classifier were jointly trained using labelled data by optimising the cross-entropy loss. In Supervised-single, a separate model was trained on the labelled data of each anchor device. On the other hand, Supervised-multi trained a common model using labelled data from all devices present in the dataset.

Semi-supervised learning

As an example of semi-supervised learning, we used two Autoencoder [Baldi, 2012] baselines: AutoEncoder-single and AutoEncoder-multi. In both of these baselines, as described in Section 2.2.1, the feature extractor acted as an 'encoder' that converted unlabelled input samples into feature embeddings. We added a separate 'decoder' neural network which did the inverse task, i.e., it mapped the feature embeddings back to the input samples. The encoder and decoder together formed the Autoencoder (AE) and were trained by minimising the mean squared error between the input data and the reconstructed data. In AutoEncoder-single, a separate AE was trained for each anchor device, whereas in AutoEncoder-multi, a common AE was trained using data from all devices. After the AE training converged, we discarded the decoder and used the trained encoder as our feature extractor $f(.)$. Subsequently, $f(.)$ was fine-tuned on the labelled data from the anchor device.

Self-supervised learning

To compare the performance of ColloSSL with a state-of-the-art self-supervised learning (SSL) technique, we again adopted the Multi-task SSL technique proposed by [Saeed et al., 2019], as previously described in Section 2.2.6 and Section 3.2.3. Note that although there are other SSL techniques proposed for HAR, we chose the work by [Saeed et al., 2019] as a baseline, because it also applies transformations to the sensor data values, thus making it a fairer comparison against ColloSSL.

We have not compared ColloSSL against our SelfHAR pipeline, as introduced in Chapter 3, because of the difference in data assumption and focus: ColloSSL leverages unlabelled but time-aligned data from multiple devices, while SelfHAR focuses on large-scale unlabelled datasets from unrelated devices.

4.4.3 Data preparation

Similar to our previous work (as discussed in Section 3.3.2), the accelerometer and gyroscope traces were segmented into time windows of 3 seconds for RealWorld and 2 seconds for Opportunity and PAMAP2 datasets, but without any overlap. These window sizes were different from our previous work and were chosen based on prior explorations with these particular datasets [Liono et al., 2016, Chang et al., 2020]. Finally, the dataset was normalised to be in the range of -1 and 1.

4.4.4 Experimental setup

Our training setup was implemented in Tensorflow 2.0 [Abadi et al., 2016]. We performed hyperparameter tuning using the Tensorflow HParams API and arrived at the following training hyperparameters: {ColloSSL learning rate: 1e-5, fine-tuning and supervised learning rate: 1e-3, $\tau = 0.05$, batch size = 512}.

In ColloSSL, even though we used unlabelled data from multiple devices to train the feature extractor, our evaluation was always done on a single anchor device (e.g., chest-worn IMU for RealWorld). We divided the participants into multiple groups for testing. The number of groups for RealWorld, Opportunity, and PAMAP2 was set to 5, 4, and 4, respectively, resulting in testing data coming from 20% to 25% of users, similar to that in our previous work (as described in Section 3.3.2), but we conducted leave-one-group-out cross-validation instead in this work. More specifically, we trained the feature extractor using unlabelled data from all groups except a *held-out* group. Thereafter, the feature extractor along with the classification head was fine-tuned on a labelled dataset from the anchor device. Finally, the fine-tuned model was tested on the data from the *held-out* group.

As we have observed general class imbalance in the datasets used for evaluation, and the relatively lower number of activities in some of the datasets, we used macro F_1 score as the main performance metric, which is considered a reasonable evaluation strategy for imbalanced datasets [Plötz, 2021], and this allowed us to demonstrate how the model performs when each class is considered equal.

4.5 Evaluation and discussion

We evaluate ColloSSL following the aforementioned evaluation protocol, and our key results include:

- ColloSSL outperforms the fully-supervised learning baseline in the low-data regime. In 15 out of the 18 anchor devices, ColloSSL trained with 10% or 25% of the labelled data achieves a higher F_1 score than the fully-supervised model trained with 100% labelled data.
- ColloSSL also outperforms various HAR baselines in terms of recognition performance. When the same amount of labelled data is used, ColloSSL has an absolute increase of 7.9% in the F_1 score, compared to the best-performing baseline.
- Through visualisation of t-SNE plots and saliency maps, we show that ColloSSL generates well-separable and meaningful feature embeddings across classes.
- ColloSSL is robust to temporal misalignment of data from multiple devices. Less than ±0.006 difference in the F_1 score is observed when up to 0.5s and less than ±0.01 difference when up to 3s of time synchronisation errors are introduced in the devices.

4.5.1 Performance in the low-data regime

We evaluate the HAR performance of ColloSSL against baseline techniques in two aspects. First, we study whether our solution performs on par compared to the baselines in the low-data regime, i.e., whether ColloSSL shows comparable performance with a small amount of labelled data. Second, we investigate whether our solution outperforms the baselines in terms of recognition accuracy, with the same data availability, i.e., when all baselines are trained with the same amount of labelled data.

To study the low-data regime, we fine-tune ColloSSL and other baselines except for Supervised-single, using 10%, 25%, 50%, 75%, and 100% of the available labelled data from the anchor device. The fine-tuned models are then evaluated on the users from the held-out validation group. Supervised-single is used as a reference point, thus trained using 100% of the training data. Note that the labelled training data available in our datasets for each anchor device (on average) is as follows: RealWorld: 1027 windowed samples (approximately 51 minutes), Opportunity: 3014 samples (approximately 100 minutes), PAMAP2-Locomotion: 1280 samples (approximately 42 minutes), and PAMAP2-ADL: 5709 samples (approximately 190 minutes).

Table 4.2 shows the classification performance for the various anchor devices, averaged over all validation groups in a leave-one-group-out evaluation. More specifically, we report the minimum percentage of labelled data required by each technique to surpass the

Table 4.2: Classification performance: average of macro F_1 scores and the minimum percentage of labelled data required to outperform the fully-supervised model (Supervised-single). For each anchor device, bold numbers represent the highest F_1 score, and red numbers indicate the technique which requires the least amount of labelled data to outperform Supervised-single. When a technique does not outperform Supervised-single, we denote its best-achieved performance with an asterisk.

Dataset (anchor)	Supervised-single	Random	Supervised-multi	AutoEncoder-single	AutoEncoder-multi	Multi-task SSL	ColloSSL
RealWorld							
forearm	0.732 (100%)	0.253 (50%)*	0.495 (100%)*	0.723 (100%)*	0.739 (75%)	0.734 (50%)	**0.767** (25%)
head	0.643 (100%)	0.211 (25%)*	0.537 (100%)*	0.647 (100%)	0.646 (25%)	0.670 (25%)	**0.690** (10%)
shin	0.781 (100%)	0.375 (100%)*	0.628 (100%)*	0.784 (10%)	0.765 (75%)*	**0.810** (10%)	**0.810** (10%)
chest	0.715 (100%)	0.228 (50%)*	0.650 (100%)*	0.478 (75%)*	0.720 (25%)	**0.722** (10%)	0.716 (25%)
thigh	0.701 (100%)	0.283 (100%)*	0.586 (100%)*	0.695 (75%)*	0.656 (25%)*	0.675 (75%)*	0.690 (25%)*
upper arm	0.726 (100%)	0.268 (75%)*	0.595 (100%)*	0.739 (75%)	0.708 (100%)*	**0.753** (10%)	0.740 (25%)
waist	0.745 (100%)	0.297 (25%)*	0.674 (100%)*	0.582 (75%)*	0.770 (10%)	0.778 (10%)	**0.781** (10%)
Opportunity							
back	0.439 (100%)	0.164 (50%)*	0.253 (25%)*	0.446 (10%)	0.445 (50%)	0.380 (25%)*	**0.556** (10%)
lla	0.370 (100%)	0.197 (100%)*	0.398 (25%)	0.386 (25%)	0.375 (25%)	0.374 (100%)	**0.516** (10%)
left shoe	0.391 (100%)	0.164 (10%)*	0.396 (75%)	0.282 (100%)*	0.172 (25%)*	0.164 (100%)*	**0.416** (25%)
right shoe	0.378 (100%)	0.164 (10%)*	0.392 (25%)	0.265 (100%)*	0.166 (50%)*	0.183 (100%)*	**0.402** (10%)
rua	0.416 (100%)	0.164 (10%)*	0.293 (100%)*	0.447 (10%)	0.375 (10%)*	0.277 (10%)*	**0.538** (10%)
PAMAP2-Locomotion							
ankle	0.731 (100%)	0.609 (50%)*	0.589 (10%)*	0.651 (10%)*	0.770 (10%)	0.774 (50%)	**0.784** (100%)
chest	0.654 (100%)	0.295 (50%)*	0.738 (10%)	0.669 (75%)	0.655 (10%)	0.730 (10%)	**0.741** (10%)
hand	0.723 (100%)	0.496 (25%)*	0.731 (25%)	0.723 (100%)	0.750 (10%)	**0.791** (10%)	0.740 (10%)
PAMAP2-ADL							
ankle	0.550 (100%)	0.262 (100%)*	0.548 (100%)*	0.560 (25%)	0.489 (100%)*	0.567 (50%)	**0.578** (25%)
chest	0.640 (100%)	0.156 (100%)*	0.640 (50%)	0.623 (25%)	0.607 (75%)*	0.615 (100%)*	**0.651** (50%)
hand	0.575 (100%)	0.208 (50%)*	0.585 (25%)	0.577 (75%)	0.586 (50%)	0.596 (50%)	**0.617** (25%)

performance of Supervised-single (in parenthesis), and the corresponding macro F_1 score averaged over all validation groups. In case a technique does not surpass the performance of Supervised-single, we report its best performing F_1 score and labelled data percentage.

Our results confirm the data efficiency of ColloSSL. In 15 out of the 18 anchor devices, including those for ADL recognition, ColloSSL with 10% or 25% of labelled data achieves a higher F_1 score than Supervised-single trained with 100% labelled data. In the remaining three cases, ColloSSL shows a comparable F_1 score with 25% of labelled data when evaluated at the thigh-worn device in RealWorld (ColloSSL: 0.690, Supervised-single: 0.701), a higher F_1 score with 100% of labelled data when evaluated at the ankle-worn device in PAMAP2-Locomotion (ColloSSL: 0.784, Supervised-single: 0.731), and a comparable F_1 score with 50% of labelled data evaluated at the chest-worn device in PAMAP2-ADL (ColloSSL: 0.651, Supervised-single: 0.640).

Table 4.2 also shows (in the red font) the technique which requires the least amount of labelled data to surpass Supervised-single. We observe that ColloSSL generally performs better compared with other semi-supervised approaches (AutoEncoder-single and

AutoEncoder-multi) and the self-supervised approach (Multi-task SSL). More specifically, in 13 out of 18 anchor devices, ColloSSL used the lowest percentage of labelled data across the baselines. This is remarkable considering that AutoEncoder-single, AutoEncoder-multi, and Multi-task SSL only win at 5, 4, and 6 anchor devices; note that there can be multiple winners.

Finally, Table 4.2 shows (in bold font) the technique which provides the highest performance in the low-data regime. Here ColloSSL has the highest F_1 score in 14 out of 18 anchor devices across all techniques. Multi-task SSL also outperformed the supervised baseline in most cases, and outperformed ColloSSL in 3 of the remaining scenarios. This could be attributed to the data efficiency of self-supervised methods, and in certain scenarios, pre-text tasks based on specific data transformations could offer the right data diversity for training a feature extractor. However, for the Opportunity dataset, there are several cases where the Multi-task SSL baseline failed to outperform the supervised baseline. One possible reason for this is that the effectiveness of SSL methods which rely on manual data transformations can be highly dependent on the dataset and the specific transformations used by the technique [Chen et al., 2020a]. It is possible that the manual transformations employed by this baseline were not optimal for Opportunity, which resulted in poor recognition accuracy. Instead, ColloSSL does not define any manual transformations on the datasets and leverages natural transformations present in the data, and therefore it is more robust to dataset variations.

Figure 4.4 provides further insights into these findings by plotting the performance of various techniques when they are trained or fine-tuned with different percentages of labelled data from the anchor device. We present two important findings. First, regardless of the percentage of labelled data used for fine-tuning, ColloSSL generally outperforms the baselines. This shows that our design of device selection, contrastive sampling, and group contrastive loss contributes to enhancing the performance on HAR. Second, we can again observe that ColloSSL outperforms the fully-supervised model (Supervised-single) with much less labelled data, including for PAMAP2-ADL, which contains more complex activities of daily living.

4.5.2 Comparison of recognition performance with baselines

We now compare the classification performance of ColloSSL against various baseline techniques when sufficient labelled data is available. Here, we use 100% of the labelled training data available from the anchor device for fine-tuning ColloSSL, AutoEncoder-single, AutoEncoder-multi, and Multi-task SSL, and for training Supervised-single and Supervised-multi. Then, we evaluate these techniques on the anchor device from the held-out group. A hyperparameter search on training parameters was conducted for all pipelines to ensure optimal performance.

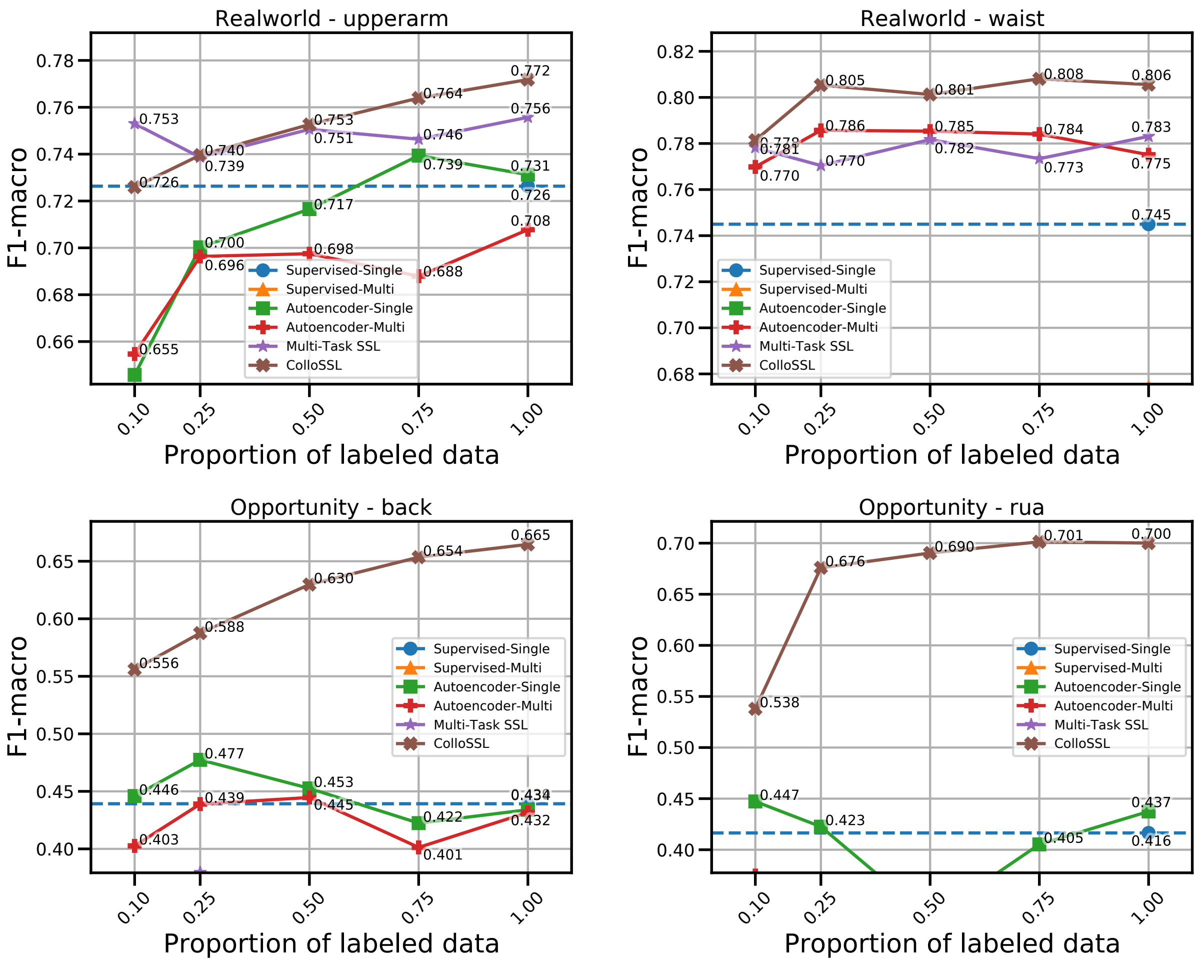


Figure 4.4: Assessing the classification performance of ColloSSL and baselines across different percentages of labelled data. These plots focus on the performance of ColloSSL and baselines which achieved similar levels of performance. Some baselines have poor performance, and therefore, are not shown in the plot. Please refer to Table 4.2 for in-depth results.

Table 4.3 shows the mean and standard deviation of macro F_1 scores for different anchor devices. On average, the results show that ColloSSL outperforms baseline techniques for all datasets we used. ColloSSL has an absolute increase of around 7.9% in the F_1 score over the average of all anchor devices across all datasets, compared to the best-performing baseline, Supervised-single. We also observe that ColloSSL outperforms Multi-task SSL (Transformation Discrimination), a state-of-the-art self-supervised learning technique for HAR for all except one anchor device. This validates our design choice of leveraging natural transformations from the TSMDS settings for self-supervised contrastive learning. Furthermore, our method outperforms the best-performing baseline by 4% in F_1 score in absolute terms in PAMAP2-ADL, which demonstrates that our proposed method can offer performance gain in simpler locomotion recognition, as well as more complex ADL

Table 4.3: Comparison of classification performance (average and standard deviation of macro F_1 scores) for different anchor devices, when 100% of the labelled data is available for fine-tuning. (Abbreviations for devices: lla – left lower arm, rua – right upper arm)

Dataset (anchor)	Random	Supervised-single	Supervised-multi	AutoEncoder-single	AutoEncoder-multi	Multi-task SSL	ColloSSL
RealWorld							
forearm	0.248 (0.028)	0.732 (0.065)	0.495 (0.039)	0.723 (0.045)	0.718 (0.064)	0.738 (0.057)	**0.774 (0.053)**
head	0.123 (0.028)	0.643 (0.055)	0.537 (0.031)	0.647 (0.071)	0.627 (0.061)	0.663 (0.026)	**0.730 (0.046)**
shin	0.375 (0.045)	0.781 (0.044)	0.628 (0.060)	0.799 (0.064)	0.761 (0.078)	0.785 (0.052)	**0.806 (0.103)**
chest	0.135 (0.030)	0.715 (0.104)	0.650 (0.054)	0.461 (0.127)	**0.729 (0.09)**	0.708 (0.061)	0.720 (0.095)
thigh	0.283 (0.024)	**0.701 (0.11)**	0.586 (0.022)	0.670 (0.061)	0.616 (0.088)	0.651 (0.120)	0.679 (0.101)
upper arm	0.126 (0.028)	0.726 (0.090)	0.595 (0.019)	0.731 (0.066)	0.708 (0.063)	0.756 (0.084)	**0.772 (0.042)**
waist	0.157 (0.049)	0.745 (0.127)	0.674 (0.042)	0.579 (0.167)	0.775 (0.051)	0.783 (0.102)	**0.806 (0.070)**
Average	0.207 (0.090)	0.720 (0.039)	0.595 (0.058)	0.659 (0.103)	0.705 (0.057)	0.726 (0.050)	**0.755 (0.044)**
Opportunity							
back	0.164 (0.010)	0.439 (0.092)	0.217 (0.012)	0.434 (0.114)	0.432 (0.107)	0.355 (0.076)	**0.665 (0.134)**
lla	0.197 (0.050)	0.370 (0.013)	0.396 (0.082)	0.458 (0.028)	0.369 (0.016)	0.374 (0.011)	**0.553 (0.018)**
left shoe	0.164 (0.009)	0.391 (0.043)	0.394 (0.046)	0.282 (0.073)	0.171 (0.010)	0.164 (0.009)	**0.443 (0.040)**
right shoe	0.164 (0.009)	0.378 (0.024)	0.354 (0.032)	0.265 (0.056)	0.164 (0.009)	0.183 (0.011)	**0.448 (0.026)**
rua	0.164 (0.009)	0.416 (0.060)	0.293 (0.068)	0.437 (0.126)	0.277 (0.058)	0.185 (0.034)	**0.700 (0.131)**
Average	0.171 (0.013)	0.399 (0.025)	0.331 (0.068)	0.375 (0.084)	0.283 (0.106)	0.252 (0.092)	**0.562 (0.107)**
PAMAP2-Locomotion							
ankle	0.558 (0.115)	0.731 (0.100)	0.558 (0.072)	0.635 (0.012)	0.754 (0.081)	0.720 (0.095)	**0.784 (0.088)**
chest	0.160 (0.052)	0.654 (0.136)	0.680 (0.082)	0.687 (0.120)	0.639 (0.098)	0.716 (0.141)	**0.742 (0.112)**
hand	0.397 (0.180)	0.723 (0.111)	0.738 (0.092)	0.723 (0.105)	0.729 (0.088)	**0.777 (0.065)**	0.737 (0.078)
Average	0.372 (0.163)	0.703 (0.121)	0.659 (0.111)	0.682 (0.099)	0.708 (0.102)	0.738 (0.109)	**0.754 (0.097)**
PAMAP2-ADL							
ankle	0.262 (0.038)	0.550 (0.124)	0.548 (0.135)	0.566 (0.090)	0.489 (0.151)	0.559 (0.096)	**0.646 (0.184)**
chest	0.156 (0.062)	0.640 (0.189)	0.655 (0.177)	**0.660 (0.150)**	0.606 (0.104)	0.615 (0.169)	**0.660 (0.185)**
hand	0.170 (0.030)	0.575 (0.078)	0.647 (0.090)	0.575 (0.063)	0.585 (0.095)	0.621 (0.089)	**0.664 (0.087)**
Average	0.196 (0.047)	0.588 (0.038)	0.617 (0.049)	0.601 (0.044)	0.560 (0.051)	0.598 (0.028)	**0.657 (0.008)**
Total average	0.237	0.603	0.551	0.579	0.564	0.579	**0.682**

recognition.

4.5.3 Embedding similarity and data saliency

In this section, we delve deeper to analyse the feature embeddings learned by the feature extractor and compare them between ColloSSL and fully-supervised settings. We also present saliency maps to understand how the HAR models trained in these two settings are making their predictions.

Visualising the feature space using t-SNE plots

We visualise the learned feature embeddings of ColloSSL and the baselines (Supervised-single) using t-distributed stochastic neighbour embedding (t-SNE) plots [Maaten and Hinton, 2008], similar to those in the previous chapter.

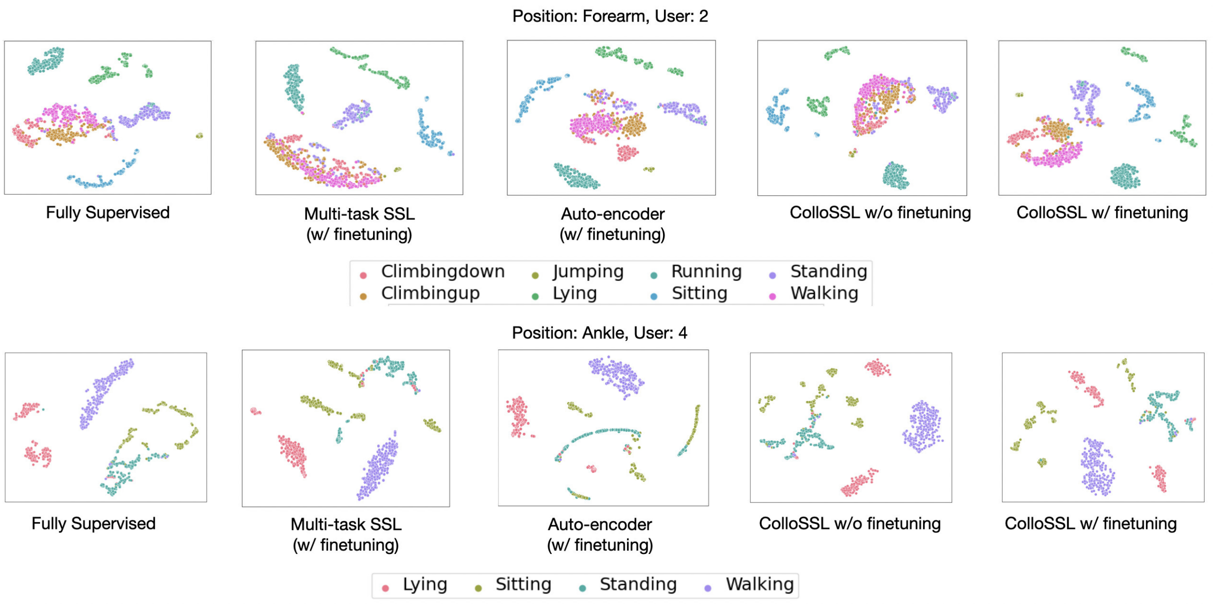
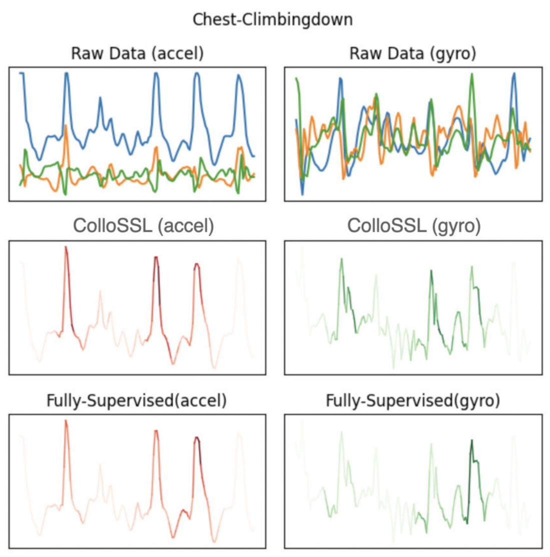


Figure 4.5: t-SNE visualisations comparing the features learned by ColloSSL and different baselines.

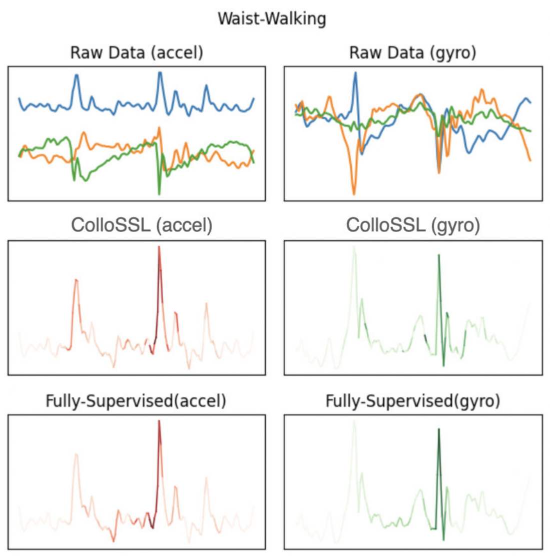
Using t-SNE, we project the 96-dimensional feature embeddings generated by the feature extractor onto a 2D space in the following settings: ColloSSL without fine-tuning, ColloSSL with fine-tuning, fully-supervised, multi-task SSL and autoencoder-single. Figure 4.5 shows the t-SNE plots for two users and two anchor devices from the RealWorld (top) and PAMAP2-Locomotion (bottom) datasets. In common, we observe that ColloSSL without fine-tuning already generates well-separable feature embeddings across classes. It implies that ColloSSL captures the semantic structure of the data very well. The class separation in the embeddings is further improved by fine-tuning with a small amount of labelled data as shown in ColloSSL with fine-tuning. We also observe that the nature of clustering learned with ColloSSL is largely comparable with those learned with a fully-supervised model trained with 100% of the labelled data. The two baselines, autoencoder and multi-task SSL, also achieve a reasonably good cluster separation. However, certain classes end up having a high overlap in their features. For example, Multi-task SSL finds it difficult to separate the *Climbing up*, *Climbing Down*, and *Walking* activities in Figure 4.5 (top).

Visualising the salient regions in the data using saliency maps

For better interpretability of our findings, we visualise saliency maps [Simonyan et al., 2014, Saeed et al., 2019] for two randomly selected data samples from the RealWorld dataset. A saliency map illustrates the regions of the data sample that have the most



(a) Climbing down activity collected from a chest-worn IMU



(b) Walking activity collected from a waist-worn IMU

Figure 4.6: Saliency maps for samples from the RealWorld dataset: (top) raw input signal, (middle) and (bottom) magnitude values computed from the input signal. The intensity of colour indicates the impact of the region on the model prediction. We observe that ColloSSL and fully supervised model show similar patterns in colour intensity. This implies that models trained with both approaches largely focus on similar regions of the data to make their predictions.

effect on a model's prediction. Our objective is to understand if the salient regions of the data remain consistent across ColloSSL and fully-supervised training.

Figure 4.6 shows the saliency maps for the selected data samples from the RealWorld dataset: a sample of the *climbing down* activity collected from the chest-worn IMU in Figure 4.6a and a sample of the *walking* activity collected from the waist-worn IMU in Figure 4.6b. We visualise the three-axis raw input data from the accelerometer and gyroscope separately in the top pane. The middle and bottom panes show the saliency maps for this input data produced by ColloSSL and fully-supervised training for the class with the highest prediction score. For ease of understanding, we only present the magnitude values of the accelerometer and gyroscope data in the saliency maps. In the middle and bottom panes, the intensity of colour indicates the impact of the region on the model prediction. The regions with strong intensity imply that they contribute to the model prediction more than those with weak intensity.

Figure 4.6 shows that ColloSSL and the fully-supervised model show a similar pattern in colour intensity, both for the accelerometer and gyroscope samples. For example, in Figure 4.6a, the periodic peaks in the accelerometer data in the x-axis (blue colour) are largely responsible for the model's prediction in both ColloSSL and fully-supervised

Table 4.4: Comparison of various device selection strategies in terms of performance (average and standard deviation of macro F_1 scores) for the PAMAP2-ADL dataset.

Anchor	Closest Positve & Random Negative	Random Selection	ColloSSL
chest	0.649 (0.175)	0.631 (0.166)	**0.662 (0.185)**
ankle	0.602 (0.122)	0.553 (0.095)	**0.646 (0.184)**
hand	0.651 (0.088)	0.634 (0.085)	**0.664 (0.087)**

settings. The takeaway from this result is that the models trained with ColloSSL and fully-supervised training largely focus on similar regions of the data to make their predictions. This confirms that ColloSSL is able to generate meaningful representations of data for HAR.

4.5.4 Analysis of the device selection strategy

In the following sections, we further perform a set of ablation studies and analyse the performance of ColloSSL under real-world constraints in sensor devices.

Our proposed device selection strategy (introduced in Section 4.3.3) uses the closest device (with the lowest MMD distance) to the anchor as the positive device and all devices as negatives, weighted by the inverse of their MMD distance to the anchor. In this ablation experiment, we compare this strategy against two baselines: (i) Random Selection and (ii) Closest Positive & Random Negative. In the Random Selection strategy, we randomly pick one positive and one negative with replacement in each batch. In the Closest Positive & Random Negative strategy, we pick the closest device with the least MMD distance to anchor but randomly choose one negative device.

Table 4.4 shows the experiment results on the PAMAP2-ADL dataset with 12 ADLs. We observe that ColloSSL outperforms the two baseline approaches. In particular, the Random Selection strategy performs the worst as it often picks positive devices which have different data distributions from the anchor device. The Closest Positive & Random Negative also has a lower performance as it does not prevent the violation of the **(N1)** criteria for negative samples as described in Section 4.3.3. This finding supports our hypothesis that using 'all' devices for negative samples serves as a hedge against the scenario when **(N1)** is violated on one of the devices.

4.5.5 Analysis of the contrastive sampling approach

We now evaluate the effect of contrastive sampling by running an ablation on the PAMAP2-ADL dataset. We compare ColloSSL's asynchronous negative sampling against its counterpart, synchronous negative sampling. Note that, positive samples are sampled syn-

Table 4.5: Comparison of the effect of contrastive sampling on performance (average and standard deviation of macro F_1 scores) for the PAMAP2-ADL dataset.

Anchor	Synchronous positive and negative	ColloSSL
chest	0.630 (0.180)	**0.662 (0.185)**
ankle	0.601 (0.101)	**0.646 (0.184)**
hand	0.639 (0.076)	**0.664 (0.087)**

Table 4.6: Comparison of the effect of weights in the multi-view contrastive loss over performance (average and standard deviation of macro F_1 scores) of ColloSSL in the PAMAP2-ADL dataset.

Anchor	ColloSSL without weights	ColloSSL
chest	0.658 (0.184)	**0.662 (0.185)**
ankle	0.601 (0.112)	**0.646 (0.184)**
hand	0.636 (0.076)	**0.664 (0.087)**

chronously in both settings, otherwise positive samples will violate characteristic **(P1)**. Table 4.5 exhibits the improvement in performance by using asynchronous negative sampling. We attribute this gain to the knowledge that synchronous samples in the TSMDS setting belong to the same class as the anchor sample. Therefore, by negatively contrasting these samples, the feature extractor is violating **(N1)** and learns suboptimal representations.

4.5.6 Analysing the role of weights in the multi-view contrastive loss

In the multi-view contrastive loss, we introduce weights as a way to differentiate the contributions between negative devices towards the optimisation objective. To investigate the effects these weights have on ColloSSL, we conducted an experiment where the weights were removed from the loss function (i.e., all devices were assigned the same weight). Table 4.6 shows the results of our experiment. We can observe that the use of the weighted loss function improves the performance of ColloSSL which supports our hypothesis that weighted negatives help in pushing closer negative embeddings further away from the anchor device **(N2)** resulting in better features.

Table 4.7: Performance (macro F_1 scores) for two anchor devices in the RealWorld dataset under different levels of sensor heterogeneity. 'None' denotes the case where no additional sensor error is added to the dataset.

	None			Low			High		
Anchor	Supervised single	Multi-task SSL	ColloSSL	Supervised single	Multi-task SSL	ColloSSL	Supervised single	Multi-task SSL	ColloSSL
forearm	0.732	0.738	**0.774**	0.744	0.762	**0.786**	0.730	0.740	**0.761**
shin	0.781	0.785	**0.806**	0.806	0.811	**0.833**	0.773	0.782	**0.800**

4.5.7 Robustness to sensor heterogeneity

In the TSMDS setting, devices placed at different body positions could be heterogeneous, in which they can come from different manufacturers or use different inertial sensors. In this section, we probe the robustness of ColloSSL to sensor heterogeneity by synthetically adding two types of heterogeneity in IMU data based on prior literature [Frosio et al., 2008, Poddar et al., 2017, Grammenos et al., 2018].

Prior research has shown that deterministic errors in IMU sensors are the prominent causes of heterogeneity in the sensor data. Deterministic errors are caused by variations in sensor components across manufacturers, imperfections introduced in the analogue circuitry of the sensor during the manufacturing process [Dey et al., 2014], or temperature differences between initial calibration and operational stages [Aggarwal et al., 2008]. Two of the major types of deterministic errors are scale factor errors and bias errors [Grammenos et al., 2018].

For this experiment, we induce different scale factors and bias errors in each IMU device in the RealWorld dataset. For each device, we sample a scale factor S from a normal distribution with $\mu = 1.0$ and $\sigma = 0.05$ (low heterogeneity) and $\sigma = 0.1$ (high heterogeneity). Similarly, we sample a bias factor B from a normal distribution with $\mu = 0.0$ and $\sigma = 0.05$ (low heterogeneity) and $\sigma = 0.1$ (high heterogeneity). Following the methodology proposed in [Grammenos et al., 2018], the two factors are introduced to the raw datasets $\{\mathbb{X}_i\}_{i=1}^{|\mathbb{D}|}$ to obtain $\mathbb{X}'_i = S \times (\mathbb{X}_i - B)$, where $\mathbb{X}'_i$ denotes the dataset with induced sensor heterogeneity for the i^{th} device. Thereafter, we run the end-to-end training pipeline of ColloSSL on the heterogeneous datasets $\{\mathbb{X}'_i\}_{i=1}^{|\mathbb{D}|}$ using the same experiment protocol as the previous experiments.

Our findings are shown in Table 4.7. For comparison, we also present the results when no additional sensor error is added to the dataset. Overall, we observe that ColloSSL is able to handle sensor heterogeneity and outperforms the fully-supervised and multi-task SSL baselines. Interestingly, we found that introducing a low level of heterogeneity to the unlabelled data improves the performance of ColloSSL and the other baselines. This finding can be explained by prior work which has shown that data augmentation helps deep neural networks learn better and more generalizable features [Mathur et al.,

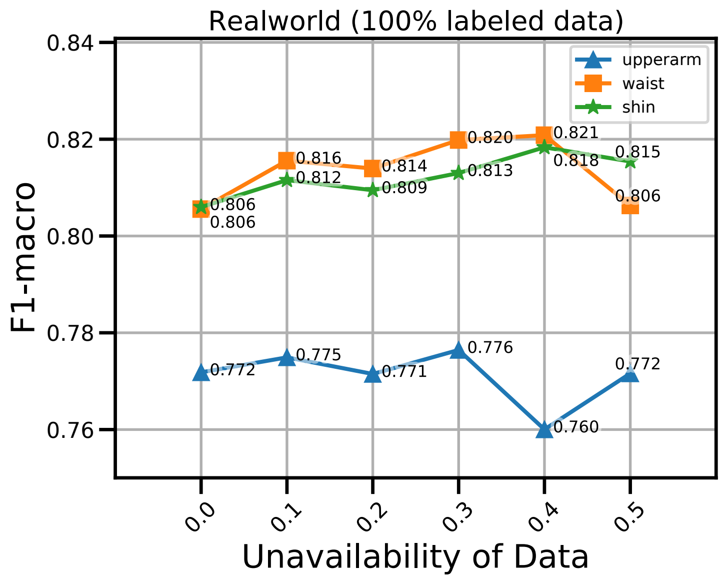


Figure 4.7: Assessing the classification performance of ColloSSL across different unavailability of devices in TSMDS setting. Note that the unavailability of each device (x-axis) is decided independently of each other. Please refer to Section 4.5.8 for more details.

2018, Park et al., 2019].

4.5.8 Robustness to missing devices

In real-world scenarios, it is often the case that not all devices are available all the time. For example, the device might run out of battery, or a user might choose to take off their earbuds during a conversation. This would result in having missing signal data from some devices in the TSMDS setting. We explore this missing data problem for ColloSSL and conduct an experiment with the RealWorld dataset. While preparing the data for this experiment, we assume that the anchor device, for which we would like to learn a prediction model, is always available. For the remaining N devices, we set the unavailability of each device with the probability, p_u= $\{0.1, 0.2, 0.3, 0.4, 0.5\}$. For example, if p_u is set to 0.1, all devices will be available in the same time window with the probability of $(1 - 0.1)^N$. More specifically, when a device is set to unavailable in a given time window, we replace its sensor data with zeros.

Figure 4.7 shows the F_1-macro values with varying availability probability values. The results show that ColloSSL is robust against changes in the availability of devices and we observe at most a 1% performance drop in our experiments due to device unavailability. This result can be explained by the design of device selection and weighted loss function in ColloSSL. Firstly, missing devices (i.e., devices with 0 data) will have a high MMD with the anchor device, and ColloSSL is likely to assign them as negative devices. Secondly, the

Table 4.8: Comparison of classification performance (average of macro F_1 scores) among different time synchronisation errors.

Anchor device	Time Synchronisation error							
(RealWorld)	0s	0.01s	0.1s	0.5s	0.75s	1.5s	2.25s	3s
waist	0.806	0.804	0.800	0.808	0.811	0.805	0.802	0.812
shin	0.806	0.809	0.809	0.805	0.807	0.813	0.811	0.815

contribution of these missing devices will be significantly down-weighted in the multi-view loss function as negative devices with high MMD distances get assigned smaller weights in training. Surprisingly, we also observe that ColloSSL with missing devices sometimes provides slightly better performance than the case with full device availability. This could be caused by the neural network considering missing data as a form of noise, which might lead to an implicit training regularisation that boosts performance.

4.5.9 Robustness to temporal misalignment

In Section 4.2, we assume that data from multiple devices in the TSMDS setting are collected in a time-aligned manner. To investigate how robust ColloSSL is to temporal misalignment between devices, we conducted an experiment with the RealWorld dataset by deliberately injecting time synchronisation errors. More specifically, we shift the timestamps of all devices in the RealWorld dataset by 0.01, 0.1, 0.5, 0.75, 1.5, 2.25, and 3 seconds, except for the anchor device.

Table 4.8 shows the F_1-macro values for two anchor devices, *waist* and *shin*. The results show that, for realistic, moderate time-sync errors ($\leq$ 0.5 seconds), there is no significant change in the performance of ColloSSL, i.e., within ± 0.006 of the F_1-macro value. For cases with high misalignment ($>$ 0.5 seconds), the change in the F_1-macro score is about ± 0.01. We conjecture that this result is caused by (a) the temporal locality of human behaviours, and (b) the ability of the feature extractor $f(.)$ to ignore moderate synchronisation errors.

4.5.10 Generalisability of the feature extractor

We further investigate the generalisability of ColloSSL: whether the feature extractor $f(.)$ trained using ColloSSL is transferable to new devices, i.e., the ones that do not participate in pre-training $f(.)$. To this end, we pre-train *ColloSSL-unseen* on unlabelled data from all devices except for one 'unseen' device. The pre-trained model is then fine-tuned and evaluated on the unseen device. For example, in the RealWorld dataset – if *head* is chosen as the unseen device, we pre-train the feature extractor on the rest of the devices to obtain ColloSSL-unseen. Then, we fine-tune ColloSSL-unseen with labelled data from *head*, and

Table 4.9: Comparison of classification performance (average and standard deviation of macro F_1 scores) between ColloSSL and ColloSSL-unseen in the RealWorld dataset.

Device for testing	ColloSSL	ColloSSL-unseen
upper arm	0.772 (0.042)	0.764 (0.063)
waist	0.806 (0.070)	0.792 (0.072)

report the classification performance using test data from *head*.

Table 4.9 compares the classification performance between ColloSSL and ColloSSL-unseen when the model is evaluated at upper arm and waist-worn devices in the RealWorld dataset. The results demonstrate that ColloSSL-unseen shows comparable performance to ColloSSL, even though the data of the unseen device is not used for pre-training the feature extractor. The decrease of F_1 score is less than 1% in both cases. This gives us an early indication that the feature extractor, $f(.)$, trained using ColloSSL is transferable across devices and can be useful for fine-tuning on unseen devices. More specifically, when a new, unseen device is added to the TSMDS setting, we can reuse the pre-trained $f(.)$ and just fine-tune it using a small amount of labelled data from the new device.

4.5.11 Discussion and limitations

In this section, we discuss the limitations of our approach and elaborate on some of the practical deployment concerns associated with our method.

Training cost of ColloSSL

Training a model using ColloSSL naturally takes more time when compared to fully-supervised learning, because of the need for pre-training on unlabelled data. However, since model training is currently done offline (e.g., on a server), it has no adverse implications for system resources on mobile or wearable devices. Further, ColloSSL does not impose any additional costs for data collection. In the TSMDS setting, multiple devices (e.g., a smartphone and a smartwatch) are already collecting sensor data related to a user's activity, and ColloSSL simply uses this unlabelled data to train a more accurate HAR model.

Runtime system cost

Although ColloSSL uses data from multiple devices to train the HAR model, it is important to note that the trained model using ColloSSL only operates on a single device at runtime, similar to any conventional HAR model. Hence, we expect that the system costs of ColloSSL, such as inference latency and energy consumption on mobile and wearable devices, are the same as a HAR model trained using supervised learning.

ColloSSL as a general framework for learning in TSMDS settings

Although we focus on applying ColloSSL to HAR with motion data, the TSMDS setting is common to other sensor modalities such as audio and vision, as shown in Figure 4.1. To apply ColloSSL to other TSMDS settings, the technical solutions (device selection, contrastive sampling, and multi-view contrastive loss) might need to be redesigned to reflect the characteristics of the corresponding sensory signals, user behaviour, and environments. However, we believe that the key idea of ColloSSL is still valid, which is to leverage natural transformations in unlabelled datasets from multiple devices to generate a supervisory signal for training.

Data privacy

ColloSSL is designed as a collaborative learning framework, in that it requires raw data from multiple devices to train a HAR model. In practice, the sensor devices owned by a user may be from different device manufacturers, and the user may not be willing to offload the raw sensor data to a centralised cloud server due to privacy and commercial reasons. We envision two potential solutions to this issue: firstly, model training can be done on a trusted edge device such as a home router and it ensures that a user's data never leaves their premises. Alternatively, federated self-supervised learning approaches [Shi et al., 2021] can be explored wherein the feature extractor is trained locally on each device and only the gradients of the feature extractor are shared to a central server for aggregation.

Extension to other SSL algorithms

This work focuses on using a contrastive learning paradigm with positive and negative samples for self-supervised learning. However, novel SSL methods that do not require negative samples have been proposed [Grill et al., 2020, Chen and He, 2021] and they were shown to outperform contrastive learning methods such as SimCLR in certain settings. Changes to the ColloSSL training pipeline would be needed to adapt to these methods.

4.6 Conclusions

In this chapter, motivated by the growing popularity of multi-device systems, we presented Collaborative Self-Supervised Learning (ColloSSL), a new method to leverage unlabelled inertial data collected from multiple body-worn devices to learn a good representation of the data. In doing so, we exploited an important characteristic of the TSMDS setting that the time-aligned data from different devices can be considered natural transformations of each other. Based on this observation, we presented a contrastive learning pipeline which intelligently gathers positive and negative samples from multiple devices, and contrasts them against a sample from the anchor device to generate a supervisory signal from unlabelled data. Our key findings are that ColloSSL outperforms both fully-supervised and semi-supervised learning techniques in the majority of the experiment settings. Secondly, ColloSSL is data-efficient and it can outperform the fully-supervised baselines using one-tenth of the labelled data in most settings. We also showed that ColloSSL could learn well-separable features from the data, and shined a light on how it makes its predictions by visualising saliency maps.

Overall, this work expanded upon the previous chapter by exploring multi-device settings and demonstrated that such settings exhibit unique opportunities for self-supervised learning and the development of data-efficient HAR methods. The training pipelines presented so far have focused on the conventional data assumption in HAR: fixed activities and stationary data distribution. However, in real-world scenarios, user behaviours and data distributions change over time. To address this, in the next chapter, we turn our focus to continual learning, where models need to adapt to changing data distributions. Specifically, we investigate how multi-task and self-supervised learning can be used to mitigate catastrophic forgetting, a major challenge in continual learning.

Chapter 5

Overcoming catastrophic forgetting with multi-task and self-supervised learning

In previous chapters we studied how multi-task and self-supervised learning can be used to develop data-efficient human activity systems using unlabelled data from one or more devices, assuming a fixed set of activities to be detected. To address the challenges in real-world settings where user behaviours change over time, we expand our focus to continual learning in this chapter, which has been discussed in Section 2.2.7. A critical challenge in adapting deep learning methods to the continual learning setting is catastrophic forgetting, in which models perform significantly worse on previously trained tasks after being trained on new data. In this chapter, we tackle this challenge in two ways. Section 5.2 looks at how multi-task learning can simulate continual learning at the initial training step, and hence sets up models to be more flexible and generalisable. Section 5.3 investigates how self-supervised learning and self-training can be leveraged in subsequent training steps to retain existing knowledge when adapting to new data.

5.1 Motivation and overview

To motivate our approach, we first look at some examples. In Section 2.2.7, we made an observation that most continual learning methods focus on the *incremental learning step* while ignoring optimisations for *base model training*. Intuitively, a more generalisable feature representation from the base model should better retain knowledge when adapting to new classes [Mittal et al., 2021]. In other words, if we can train a more transferable base model, we can alleviate catastrophic forgetting in continual learning. This is feasible since for a given network and accuracy, there exist multiple possible sets of weights from training the same architecture with different initialisations. Among these, some weights may enable better knowledge transfer, since overfitted solutions require significant changes to learn new concepts, causing loss of prior knowledge and worsening forgetting. Thus, we hypothesise that it is possible to alleviate catastrophic forgetting by training a more transferable base model.

Furthermore, existing works often use an empirical number of training epochs in the incremental learning steps. Such empirical values are often obtained by tuning with the full dataset, which is unavailable in practical continual learning settings, and these numbers vary significantly for the same dataset depending on the study. For instance, on the CIFAR-100 object recognition dataset, reported numbers of epochs vary significantly between 250 for BiC [Wu et al., 2019], 180 for WA-ADB [Zhao et al., 2020] and 70 for iCarl [Rebuffi et al., 2017]. With an increase in the number of training epochs, models tend to shift their focus to new classes due to data imbalance between old classes and new classes, which can hurt overall performance.

Following these examples, Section 5.2 focuses on improving base model training through multi-task learning, which allows the models to learn more transferable and generalisable representations to mitigate catastrophic forgetting in continual learning.

Another observation we made after the discussion in Section 2.2.7, is that although recent Continual Self-supervised Learning (CSSL) methods have seen success in adapting SSL methods for continual learning, these only address half of the problem – they focus on training a strong feature extractor continually with unlabelled data and assume that all labelled data are stored and available for fine-tuning after the final continual learning step, especially for evaluation. This violates the data assumption in continual learning where labelled data is only temporarily available, and the classifier should also learn *continually*. To address this, in Section 5.3, we put forward a general architecture in which both feature extraction and fine-tuning are performed continually from a stream of unlabelled and labelled data. Enabling continual end-to-end learning aligns better with real-world continual learning scenarios.

5.2 Improving feature generalisability with multi-task learning in class incremental learning

Multi-task learning, as we have seen in previous chapters (Section 2.2.6 and Chapter 3), involves training models on multiple related tasks in parallel while using a shared representation [Caruana, 1997]. Following our motivation in Section 5.1, we propose defining different class subsets as distinct classification tasks to train the base model. The model would be required to learn a set of weights based on varied views of the dataset, which we hypothesise would produce more transferable and generalisable representations to new classes. Moreover, the number of training epochs needed for continual learning is decided

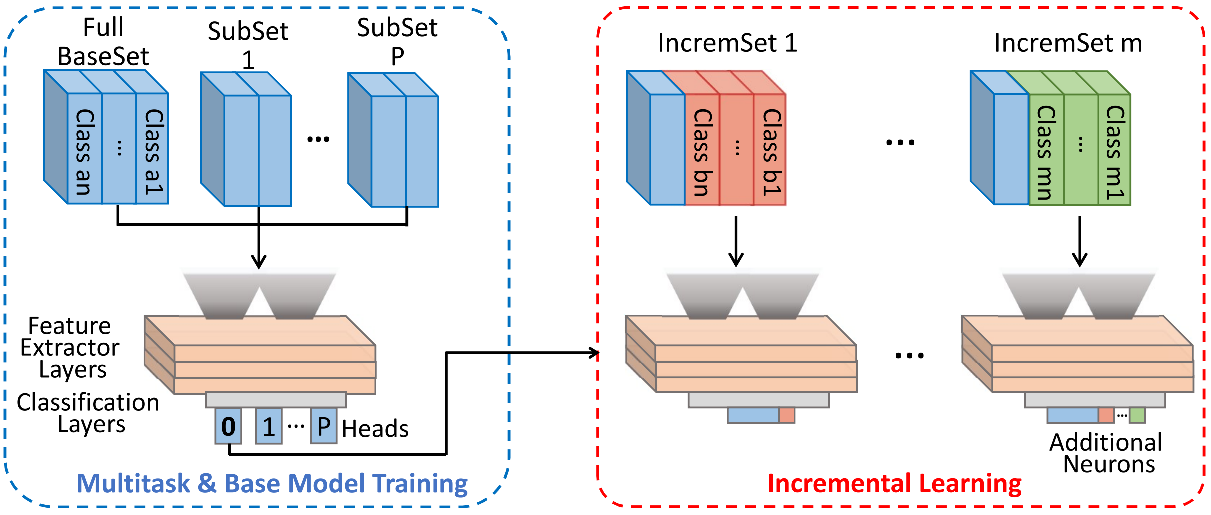


Figure 5.1: Illustration of utilising multi-task learning in class incremental learning. We introduce multi-task learning into the base model learning step in which the model is trained on multiple tasks, each corresponding to a different subset of the full base set of activities.

empirically in existing approaches. To address this, we utilise a validation set and early stopping to avoid underfitting or overfitting in practical scenarios. Experiments on the WISDM2019 (WISDM Smartphone and Smartwatch Activity and Biometrics Dataset) [Weiss, 2019], showed that our approach further improves continual accuracy by up to 6.4% and continual F_1 by up to 0.07, enabling more reliable and accurate activity recognition over time.

Furthermore, our approach is compatible with many existing approaches and provides additional gains by optimising the base model training step.

5.2.1 Approach

In this work, we introduce *multi-task learning* into class incremental learning (which has been discussed in Section 2.2.7) to improve the generalisability of feature representations. As illustrated in Figure 5.1, the base dataset is split into multiple subsets and they form a multi-task setup, where the classification of activities within each subset forms a distinct task. These tasks are trained concurrently with a shared representation. After multi-task training, the model backbone and the classifier corresponding to the full base dataset are forwarded to the incremental learning stage, where state-of-the-art continual learning techniques can be applied.

Intuition of multi-task training

Following our discussions above, multi-task learning has been used to help improve the generalisation of models by preventing models from overfitting to any particular task [Zhang and Yang, 2021]. In the context of class incremental learning, generalisability of the model is crucial, since it is necessary for the model to retain previous knowledge while learning about new classes. This motivates us to design a multi-task learning scheme that aims to simulate the incremental learning steps at the base model training stage, by splitting the dataset into different subsets.

For example, with 4 base classes: ‘Walking’, ‘Jogging’, ‘Stairs’ and ‘Sitting’, we can create different classification tasks by taking class subsets, including {‘Walking’, ‘Jogging’, ‘Stairs’}, {‘Jogging’, ‘Stairs’} and {‘Stairs’, ‘Sitting’}. The creation of multiple subsets aims to make the model find a solution which can solve all of these classification tasks at once, instead of overfitting to any particular view of the dataset. As a result, the generalisability of the base model would be improved, leading to better performance in the incremental learning stage. This particular setup is analogous to the continual learning steps, where the model is required to perform well on different sets of classes.

Multi-task creation

Important design choices arise from adopting multi-task training: (1) the number of tasks, and (2) the subset of classes that each task corresponds to. If we have N target classes in the dataset used in base model training, there are $2^N - 1$ distinct valid tasks (which equals to the number of distinct subsets excluding the empty set). If we only have a single task with all of the base classes, the learning setup degenerates to the conventional fully-supervised training setup. The number of distinct tasks grows very rapidly as N increases, so it is infeasible to train on all of them, and a design choice should be made. With the number of classes being closer to N, the task itself is more difficult, which might provide stronger supervisory signals. However, restricting all tasks to have a high number of classes reduces the diversity of tasks presented to the model, and it may not offer much help in improving the generalisability of the solution that the model converges to. In this work, we explore different configurations of tasks along two directions: (1) tasks with different numbers of classes (such as 6, 5, 4, and 3 classes for each task), and (2) those with different subsets of classes but the same number of classes (such as 6, 5, 5, and 5 classes, but each task is a distinct subset).

Learning rate scheduler

One of the most important hyperparameters affecting the amount of knowledge retention in class incremental learning is the learning rate during incremental steps, in which the models are fine-tuned. Some existing works adopt cosine annealing for training with a large starting learning rate [Mittal et al., 2021]. We observed that this large starting learning rate significantly changes the weights of the neural network, and this could make knowledge retention difficult. On the other hand, adopting a fixed small learning rate could slow down training.

Therefore, we propose a two-step fine-tuning strategy to address this. First, the majority of the upstream layers of the neural network are frozen, while the downstream layers, which include the classification layer with the newly added neurons, are trained with a relatively large learning rate. After the downstream layers, and the new neurons in particular, converge to a reasonable solution to both new and old classes, the upstream layers are unfrozen and the entire network is fine-tuned with a small learning rate. An early stopping mechanism is also adopted in our scheme because it is difficult to find the balance between overfitting and underfitting when using a fixed number of training epochs (which is commonly adopted in previous works). Furthermore, the overfitting problem is amplified in continual learning because, in the incremental learning steps, we no longer have access to the entirety of the data from previous steps, with only a small portion kept as exemplars, and overfitting to the new data particularly hurts the overall performance.

5.2.2 Experimental protocols

In this section, we introduce a set of comprehensive evaluation metrics used to evaluate our proposal and present our experimental setup.

Evaluation metrics

In order to compare the performance of different methods fairly, we define the following evaluation metrics (see Figure 5.2) that reflect different aspects of a continual learning framework, taking into account metrics defined in previous works [Díaz-Rodríguez et al., 2018, Isele and Cosgun, 2018, De Lange et al., 2021]. Here we denote the accuracy of a model on a particular task k (averaged across all classes in that task) after seeing the last sample from task t as $A_{t,k}$, and we assume that there are T tasks in total.
Final Accuracy (FA), defined as $\mathbf{FA} = \frac{1}{T}\sum_{i=1}^{T} A_{T,i}$, refers to the average model accuracy across all tasks after it has been trained on the last task T.
Continual Accuracy (CA), instead of only looking at the performance of the model at the final step, looks at the performance throughout the continual learning process, where a model might be deployed before the last task and later re-trained. We define this to be the average accuracy of the model after being trained on each task: $\mathbf{CA} = \frac{1}{T}\sum_{i=1}^{T}\left(\frac{1}{i}\sum_{j=1}^{i} A_{j,i}\right)$.
Forgetting (F) measures the extent of the performance a model has lost after training on new tasks. We calculate this by taking the difference between the highest performance

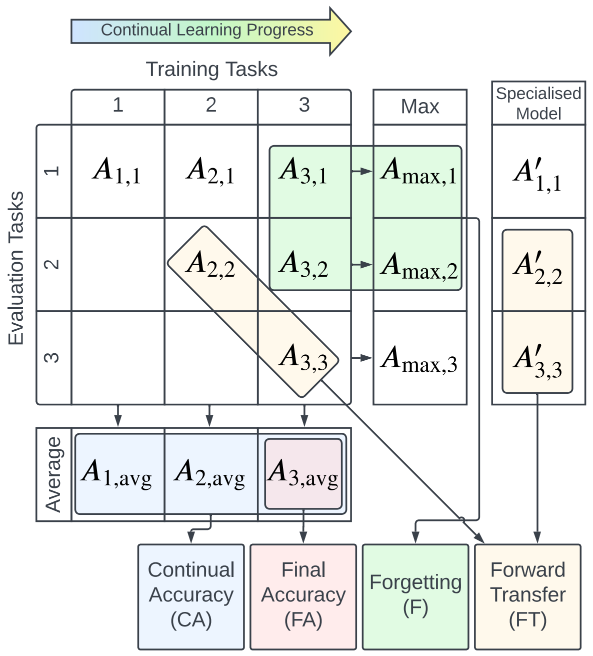


Figure 5.2: Evaluation metrics for continual learning. This figure illustrates our practical evaluation setup with regard to metrics and tasks.

of the model on a given task, and that of the model at the final step, excluding the final task: $\mathbf{F} = \frac{1}{T-1}\sum_{i=1}^{T-1}\left(A_{max,i} - A_{T,i}\right)$, where $A_{max,i} = \max_{t\in\{1,\ldots,T\}} A_{t,i}$.

Forward Transfer (FT) refers to the ability to use previously acquired knowledge when learning new tasks. Here, we define it with model deployment considerations in mind: we take the difference between the performance of the model on task k after seeing data from all tasks $i \le k$, compared to a model which has only seen data of task k. We denote the accuracy of the model which is only trained on task k as $A'_{k,k}$. FT is thus calculated by: $\mathbf{FT} = \frac{1}{T-1}\sum_{i=2}^{T}\left(A_{i,i} - A'_{i,i}\right)$. A positive value would indicate that the training on previous tasks helped the model learn the new task. It is important to note that this is a desired property for continual learning frameworks, but is usually hard to achieve.

In addition, we report macro F_1 scores for better comparison with other works in human activity recognition. In particular, we define two metrics based on F_1 scores following similar definitions outlined above: **Final $\boldsymbol{F_1}$ (FF)** and **Continual $\boldsymbol{F_1}$ (CF)**. Final F_1 is the macro F_1 score taken across all tasks after the model has been trained on the last task T, while Continual F_1 is the average of macro F_1 scores after the model is trained on each task, which captures the performance throughout the continual learning process.

This evaluation framework is set to facilitate comparisons of different properties of continual learning models in real-world applications and explorations in trade-offs between different properties such as forgetting and forward transfer. These metrics should provide guidance on selecting which method to use depending on the desired use case.

Dataset

We performed our evaluation using the WISDM2019 (WISDM Smartphone and Smartwatch Activity and Biometrics Dataset) [Weiss, 2019], which is another activity recognition dataset collected by the WISDM (Wireless Sensor Data Mining) Lab in the Department of Computer and Information Science of Fordham Unversity. The dataset contains raw accelerometer and gyroscope data from a smartwatch (LG G Watch) and a smartphone (Google Nexus 5/5x or Samsung Galaxy S5) worn by 51 subjects, who performed 18 different activities for 3 minutes each. The smartphone is placed inside the participant's pocket, while the smartwatch is worn at the dominant hand. The data was collected at a sampling rate of 20Hz for the following activities: Walking, Jogging, Stairs, Sitting, Standing, Typing, Brushing Teeth, Eating Soup, Eating Chips, Eating Pasta, Drinking from Cup, Eating Sandwich, Kicking (Soccer Ball), Playing Catch with Tennis Ball, Dribbling (Basketball), Writing, Clapping, and Folding Clothes. The accelerometer data from the smartwatch was used in this study, and we selected this dataset to evaluate our work because it has a relatively high number of activities in HAR, which makes it suitable for continual learning evaluation.

The data was similarly processed as described in Section 3.3.2, with slight modifications to reduce the window size to 384, and the windows do not overlap, resulting in a total of 9807 windows. 20%-25% of users are kept unseen for evaluation. For continual learning, we set the number of base classes and incremental classes to 6 and 3 respectively, i.e. 6 classes for the initial training step and 3 additional classes are added at each incremental step. The number of base classes is set to a higher number compared to the incremental steps so that we can evaluate the models using a more diverse set of initial training tasks (for example, we can use 6, 5, 4, 3 and 2 classes for each task for multi-task training). We set the number of labelled exemplars to be 1% for all previously seen classes except explicitly stated.

Model architecture

In this work, we used the same TPN model architecture as used in previous chapters (see Section 3.3.3 and Section 4.3.2). We used the hard parameter-sharing approach for multi-task learning, where the hidden layers for different tasks were shared and only the classification layer was trained separately for each task.

Baseline

We compared our work to the method proposed by [Mittal et al., 2021] which reported state-of-the-art results for class incremental learning. Specifically, [Mittal et al., 2021] first

Table 5.1: Time overhead when training with different task configurations. The training time is per epoch and measured in seconds, and the standard deviation is given in brackets.

Multi-task configuration	Training time (second)	Multi-task configuration	Training time (second)
[6]	0.980 (0.054)	[6,5,5]	3.002 (0.181)
[6,5]	2.070 (0.177)	[6,5,5,5]	3.339 (0.234)
[6,5,4]	2.531 (0.178)	[6,5,5,5,5]	4.124 (0.279)
[6,5,4,3]	3.767 (0.302)	[6,5,5,5,5,5]	5.824 (0.538)
[6,5,4,3,2]	4.722 (0.262)		

utilise the cross entropy ($\mathcal{L}_{\text{CE}}$) loss and knowledge distillation ($\mathcal{L}_{\text{KD}}$) loss on new classes to learn new knowledge. Then, it constructs a small but balanced exemplar set (including current incremental classes) to correct the bias and preserve the knowledge of old classes (with $\mathcal{L}_{\text{CE}}$ and $\mathcal{L}_{\text{KD}}$). For more details, please refer to the original work by [Mittal et al., 2021]. We adopted the same incremental learning strategy as in [Mittal et al., 2021] and the differences only come from base model training, i.e., single task ([Mittal et al., 2021]) vs. multi-task (ours), and the use of a two-step fine-tuning strategy as described in Section 5.2.1.

5.2.3 Evaluation and discussion

Impact of multi-task configurations

First, we systematically explored different configurations of tasks in two directions: (i) tasks with different subsets of classes, and (ii) tasks with different numbers of classes. Figure 5.3 demonstrates the change in model performance in different metrics as we vary the configuration of tasks. Figure 5.3(a)(i), (b)(i), (c)(i) and (d)(i) demonstrate that adding the initial additional task improves the performance, but adding further tasks with the same number of classes but different subsets does not lead to significant further improvements. On the other hand, Figure 5.3(a)(ii) shows that the models are able to reach the highest continual accuracy with 3 tasks ([6, 5, 4]), up to 4.3% above the single-task baseline ([6]), when each task has a different number of classes, although the performance similarly plateaus when we add more tasks. Similar conclusions can be drawn using the F_1 metrics, where the models with 3 tasks ([6, 5, 4]) outperformed the single-task baseline ([6]) by 0.046 in continual F_1 (see Figure 5.3(b)). Models with the most diverse set of tasks ([6, 5, 4, 3, 2]) achieved the highest continual F_1, but they performed minimally better than the ones with 3 tasks ([6, 5, 4]).

From Figure 5.3(e), we see that adding more tasks of the same number of classes gradually reduces forgetting, while adding tasks with different numbers of classes is able

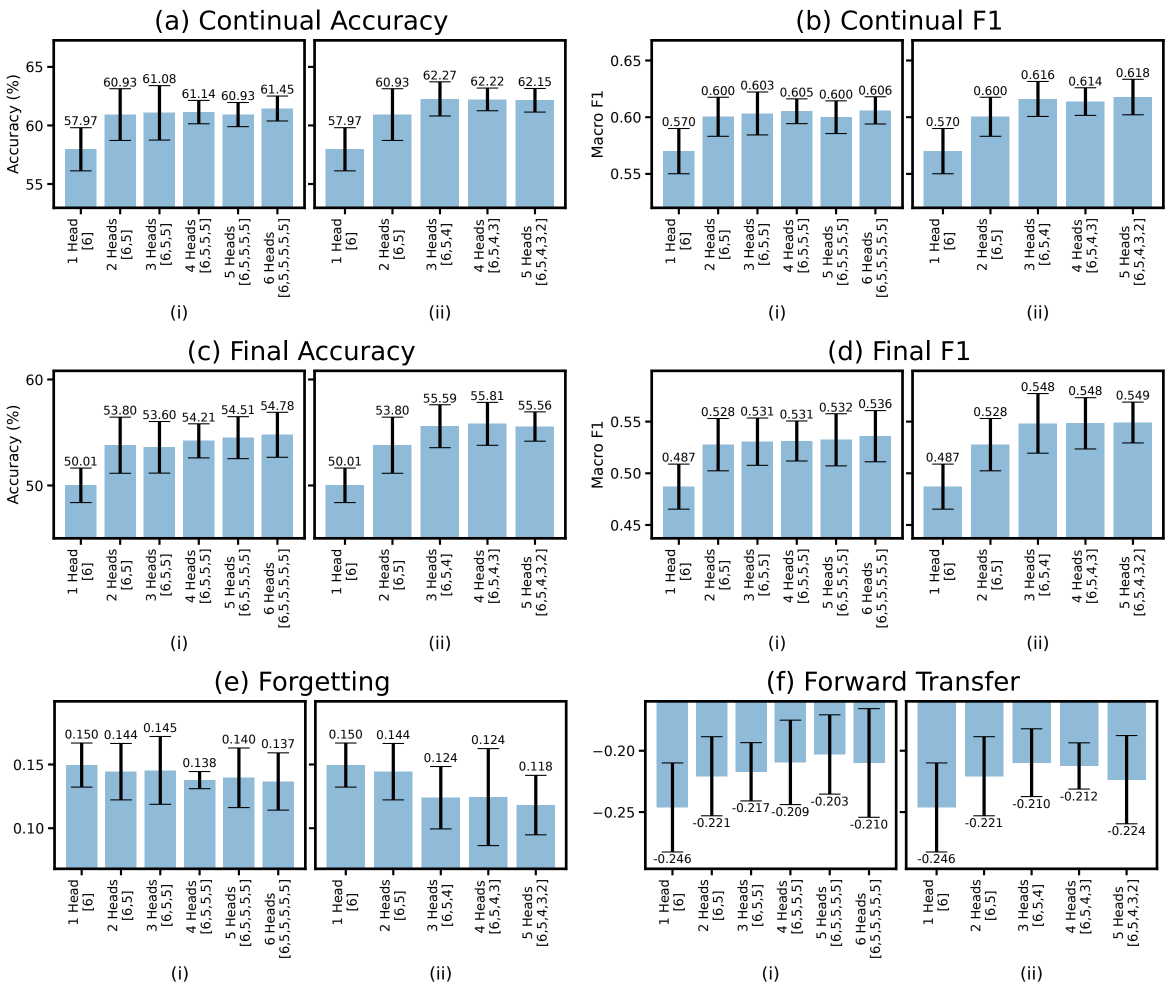


Figure 5.3: Performance comparison between different multi-task configurations. The figure shows the average (a) continual accuracy, (b) continual F_1, (c) final accuracy, (d) final F_1, (e) forgetting, (f) forward transfer of models trained with (i) tasks with different subsets of classes, and (ii) tasks with different numbers of classes. The model with only a single head corresponds to the baseline algorithm with conventional supervised training (single task).

to further reduce forgetting, reaching the minimum with 5 tasks ([6, 5, 4, 3, 2]).

As we have discussed in Section 5.2.2, Forward Transfer measures whether a model is able to perform better than a specialised model utilising previous knowledge. This is a desirable property of continual learning algorithms but is difficult to achieve, and the ability of our models to achieve Forward Transfer is shown in Figure 5.3(f). We can see that in general, our models perform worse than the specialised models on each individual task. Models trained with multi-task learning at the base learning stage are able to achieve a slightly better Forward Transfer, which follows a similar trend in other metrics discussed above.

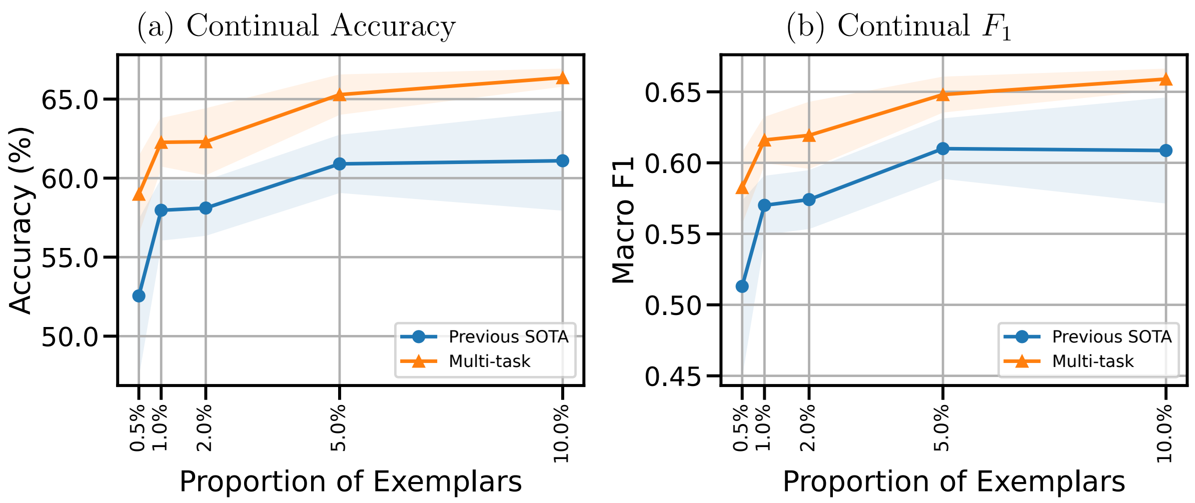


Figure 5.4: Impact of exemplar quantity on (a) continual accuracy and (b) continual F_1. The plots show the changes in performance as different percentages of labelled data are used as exemplars. The shaded regions indicate the standard deviation. Models generally achieve better performance with more exemplars.

It is interesting to note that in general, models trained with more diverse tasks (those with different numbers of classes, with results shown in Figure 5.3(ii)) are able to outperform their counterparts with more similar tasks (those with the same number of classes, shown in Figure 5.3(i)). We hypothesise that having many similar tasks may not offer much help in improving the model's generalisability, while having more diverse tasks is better at achieving that. This is because our multi-task learning objective is to allow the model to see wider and more diverse combinations of the base classes, so that it can be effectively extended to new classes during incremental learning. Consequently, if the tasks are too similar (i.e., having a high level of overlapping among classes), the training might not make the model more generalisable. For example, in the extreme case, if all the tasks contain the same set of classes, our approach degenerates to single-task learning.

In terms of time overhead during training, Table 5.1 shows a consistent increase in training time per epoch as the number of tasks increases, and when the number of classes within each task increases. Real-world deployment of this training algorithm would need to take this into account, but it is important to note that our proposal incurs an overhead only at the base training stage, which usually takes place in more powerful machines, not at the later incremental steps.

Overall, the results demonstrated that having more than a single task helps in improving the model's performance, especially when the tasks are more distinct from each other. By balancing the efficiency-accuracy trade-off, we therefore select '3 heads [6, 5, 4]' as the multi-task configuration for the rest of the evaluation.

Table 5.2: Ablation study on different losses (model accuracy). Each row of the table corresponds to the performance of models using different loss functions that are composed of $\mathcal{L}_{\text{CE}}$ (Cross-Entropy), $\mathcal{L}_{\text{KD}}$ (Knowledge Distillation), $\mathcal{L}^{\text{N}}$ (New task learning), and $\mathcal{L}^{\text{O}}$ (Old task learning). The numbers in the central columns are model accuracies (with standard deviations quoted in brackets) after being trained on different numbers of classes, corresponding to different tasks. The right-most column is the average of the central columns, which corresponds to continual accuracy.

Loss function	Number of classes 6	9	12	15	18	Average
$\mathcal{L}^{\text{N}}_{\text{CE}}$	74.79 (0.31)	51.83 (2.93)	42.71 (1.40)	36.22 (0.96)	30.21 (0.81)	47.15
$\mathcal{L}^{\text{N}}_{\text{CE}} + \mathcal{L}^{\text{O}}_{\text{KD}}$	74.79 (0.31)	57.04 (1.09)	52.09 (1.31)	46.76 (2.17)	38.81 (2.10)	53.90
$\mathcal{L}^{\text{N}}_{\text{CE}} + \mathcal{L}^{\text{N}}_{\text{KD}}$	74.79 (0.31)	57.19 (0.49)	52.09 (1.21)	46.87 (1.54)	38.92 (1.35)	53.97
$\mathcal{L}^{\text{N}}_{\text{CE}} + \mathcal{L}^{\text{N}}_{\text{KD}} + \mathcal{L}^{\text{O}}_{\text{KD}}$	74.79 (0.31)	57.55 (0.66)	51.69 (1.52)	46.18 (2.33)	38.79 (1.97)	53.80
$\mathcal{L}^{\text{N}}_{\text{CE}} + \mathcal{L}^{\text{O}}_{\text{CE}}$	74.79 (0.31)	61.02 (3.93)	57.32 (3.90)	52.69 (3.76)	52.05 (2.25)	59.57
$\mathcal{L}^{\text{N}}_{\text{CE}} + \mathcal{L}^{\text{O}}_{\text{CE}} + \mathcal{L}^{\text{O}}_{\text{KD}}$	74.79 (0.31)	62.30 (3.99)	58.83 (4.31)	54.81 (3.16)	53.66 (1.89)	60.88
$\mathcal{L}^{\text{N}}_{\text{CE}} + \mathcal{L}^{\text{O}}_{\text{CE}} + \mathcal{L}^{\text{N}}_{\text{KD}}$	74.79 (0.31)	62.42 (4.68)	59.41 (4.14)	55.21 (3.56)	54.43 (2.55)	61.25
$\mathcal{L}^{\text{N}}_{\text{CE}} + \mathcal{L}^{\text{O}}_{\text{CE}} + \mathcal{L}^{\text{N}}_{\text{KD}} + \mathcal{L}^{\text{O}}_{\text{KD}}$	74.79 (0.31)	**63.15** (2.36)	**59.69** (1.86)	**58.11** (1.70)	**55.59** (2.03)	**62.27**

Impact of exemplar quantity

As the number of exemplars available at the incremental learning stage can have a significant impact on the performance, we further conducted a set of experiments with varying quantities of exemplars: 0.5%, 1.0%, 2.0%, 5.0% and 10.0% of the labelled samples from previous tasks. Figure 5.4 shows the continual accuracy and continual F_1 of models across different amounts of exemplars. We can observe a general trend of improving performance as the number of exemplars increases, although with a diminishing effect. The performance of models starts to plateau with around 5.0% of the labelled data.

Compared to the state-of-the-art method, which uses a single task at base model learning, our method is able to consistently outperform it across different amounts of exemplars, with improvements of up to 6.4% in continual accuracy and 0.070 in continual F_1. This demonstrates efficiency in utilising exemplars: models trained with multi-task

Table 5.3: Ablation study on different losses (F_1 score). Each row of the table corresponds to the performance of models using different loss functions that are composed of $\mathcal{L}_{\mathrm{CE}}$ (Cross-Entropy), $\mathcal{L}_{\mathrm{KD}}$ (Knowledge Distillation), $\mathcal{L}^{\mathrm{N}}$ (New task learning), and $\mathcal{L}^{\mathrm{O}}$ (Old task learning). The numbers in the central columns are macro F_1 scores of the models (with standard deviations quoted in brackets) after being trained on different numbers of classes, corresponding to different tasks. The right-most column is the average of the central columns, which corresponds to continual F_1.

	Number of classes					
Loss function	6	9	12	15	18	Average
$\mathcal{L}^{\mathrm{N}}_{\mathrm{CE}}$	0.750 (0.003)	0.462 (0.042)	0.367 (0.018)	0.309 (0.026)	0.231 (0.021)	0.424
$\mathcal{L}^{\mathrm{N}}_{\mathrm{CE}} + \mathcal{L}^{\mathrm{O}}_{\mathrm{KD}}$	0.750 (0.003)	0.546 (0.010)	0.474 (0.019)	0.447 (0.027)	0.337 (0.013)	0.511
$\mathcal{L}^{\mathrm{N}}_{\mathrm{CE}} + \mathcal{L}^{\mathrm{N}}_{\mathrm{KD}}$	0.750 (0.003)	0.546 (0.010)	0.479 (0.013)	0.447 (0.012)	0.334 (0.011)	0.511
$\mathcal{L}^{\mathrm{N}}_{\mathrm{CE}} + \mathcal{L}^{\mathrm{N}}_{\mathrm{KD}} + \mathcal{L}^{\mathrm{O}}_{\mathrm{KD}}$	0.750 (0.003)	0.550 (0.006)	0.474 (0.012)	0.439 (0.019)	0.334 (0.013)	0.509
$\mathcal{L}^{\mathrm{N}}_{\mathrm{CE}} + \mathcal{L}^{\mathrm{O}}_{\mathrm{CE}}$	0.750 (0.003)	0.596 (0.035)	0.566 (0.049)	0.518 (0.038)	0.513 (0.027)	0.589
$\mathcal{L}^{\mathrm{N}}_{\mathrm{CE}} + \mathcal{L}^{\mathrm{O}}_{\mathrm{CE}} + \mathcal{L}^{\mathrm{O}}_{\mathrm{KD}}$	0.750 (0.003)	0.613 (0.033)	0.577 (0.048)	0.537 (0.023)	0.532 (0.013)	0.602
$\mathcal{L}^{\mathrm{N}}_{\mathrm{CE}} + \mathcal{L}^{\mathrm{O}}_{\mathrm{CE}} + \mathcal{L}^{\mathrm{N}}_{\mathrm{KD}}$	0.750 (0.003)	0.610 (0.043)	0.579 (0.046)	0.537 (0.031)	0.539 (0.025)	0.603
$\mathcal{L}^{\mathrm{N}}_{\mathrm{CE}} + \mathcal{L}^{\mathrm{O}}_{\mathrm{CE}} + \mathcal{L}^{\mathrm{N}}_{\mathrm{KD}} + \mathcal{L}^{\mathrm{O}}_{\mathrm{KD}}$	0.750 (0.003)	**0.625** (0.018)	**0.592** (0.015)	**0.565** (0.023)	**0.548** (0.029)	**0.616**

learning using 1.0% of data as exemplars achieve a similar level of performance to that with single-task learning using 5.0% or 10.0% of data as exemplars.

Impact of losses

To understand which technique contributes the most to incremental learning performance, we created different variations of the approach by an ablation study on different losses. Table 5.2 and Table 5.3 present the model performance at different incremental steps and the overall continual performance for incremental learning. We can observe that there is a significant drop in performance when there is no loss term dedicated to old task learning (comparing $\mathcal{L}^{\mathrm{N}}_{\mathrm{CE}}$ and $\mathcal{L}^{\mathrm{N}}_{\mathrm{CE}} + \mathcal{L}^{\mathrm{O}}_{\mathrm{KD}}$, or $\mathcal{L}^{\mathrm{N}}_{\mathrm{CE}}$ and $\mathcal{L}^{\mathrm{N}}_{\mathrm{CE}} + \mathcal{L}^{\mathrm{O}}_{\mathrm{CE}}$), and additional knowledge distillation losses only improve the performance slightly. Overall, the full combination of losses dedicated to current task learning and knowledge retention, as well as to new and old classes, is able to achieve the best performance.

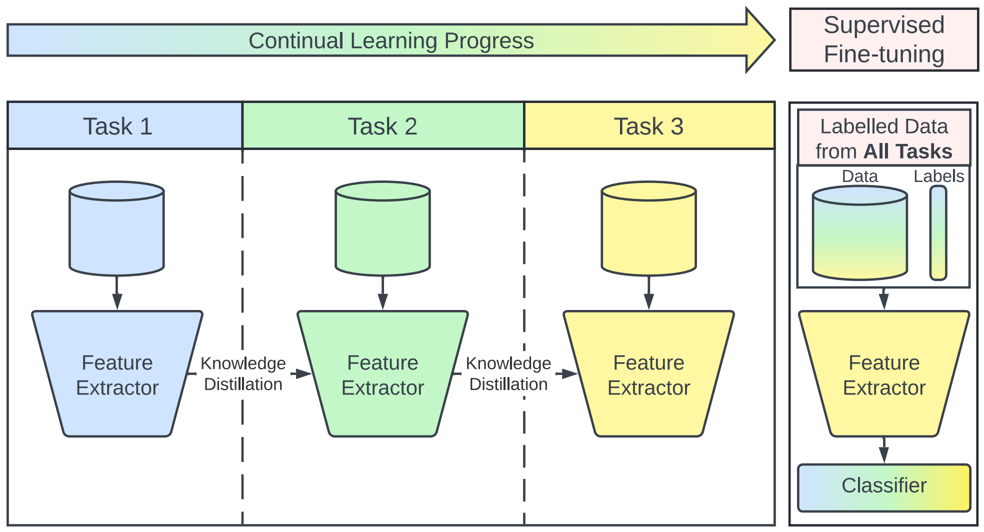
5.3 Practical self-supervised continual learning with continual fine-tuning

So far, we have seen how multi-task learning can improve model generalisability during base model training. For incremental steps, existing works mostly focus on supervised approaches using ideas such as regularisation, replay, or parameter-isolation [Kirkpatrick et al., 2017, Shin et al., 2017, Serra et al., 2018, Mittal et al., 2021] (as discussed in Section 2.2.7). Although the recent work by [Fini et al., 2022] has seen success in repurposing self-supervised learning objectives for continual learning in the area of Continual Self-Supervised Learning (CSSL), following our discussion in Section 5.1, we argue that this violates some of the core continual learning assumptions where limited to no data from earlier tasks should be available. This is because although the feature extractor learns continually, works in CSSL assume that labelled data for all classes is retained for classifier fine-tuning after the feature extractor is trained (see Figure 5.5(a)). We argue that in a more practical scenario, the classifier should be trained together with the feature extractor (see Figure 5.5(b)) because labelled data is more likely to be available during task training, rather than after the feature extractor has been trained.

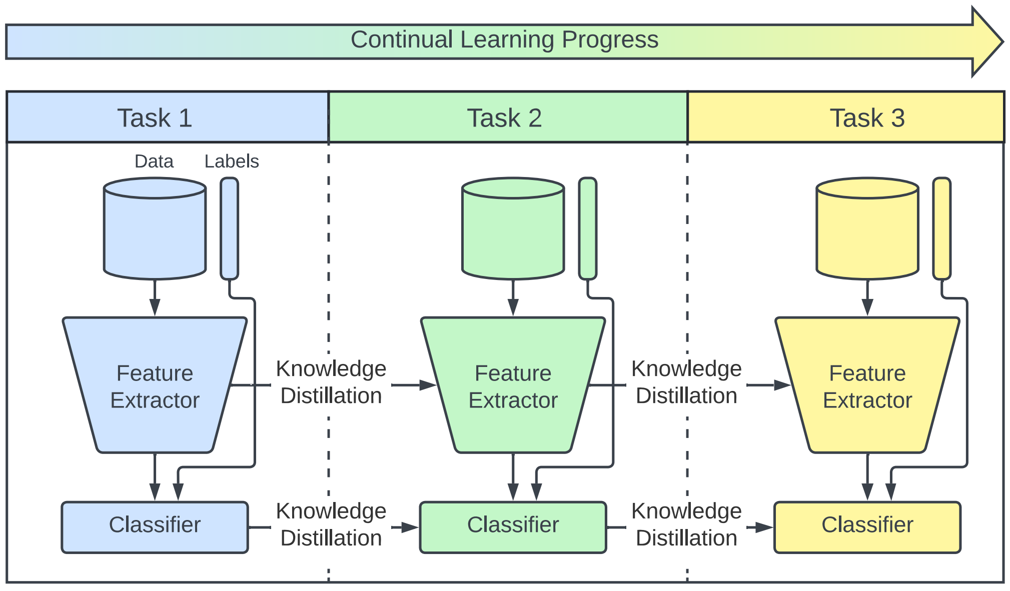
To address this, we introduce a new CSSL architecture designed to work under a more practical data assumption, where end-to-end training is performed using the continuous arrival of unlabelled and labelled data. This architecture strikes a balance between (i) training the feature extractor and fine-tuning, and (ii) combating catastrophic forgetting and learning from new data by employing a novel loss function that is specifically designed for these objectives.

We demonstrate that our proposed training pipeline outperforms previous works that focus on the self-supervised learning aspect, in terms of their overall performance and the ability to retain knowledge after training on every task. Our proposal improves performance by up to 10.6% in continual accuracy (in absolute terms), up to 0.141 in continual F_1, and up to 20.6% reduction in forgetting. The main contributions of this work are:

1. **Practical continual self-supervised learning framework.** We proposed a continual learning framework that allows model deployment at any point during the continual learning process with a functional classifier. It can leverage both unlabelled and labelled data in training the feature extractor and classifier instead of only using one type of data, in a carefully designed loss function and distillation mechanism, thus allowing higher flexibility in terms of storage requirements and in accommodating privacy concerns, which is important in real-world applications.



(a) Self-supervised Continual Learning



(b) Self-supervised Continual Learning with Continual Fine-tuning

Figure 5.5: Self-supervised Continual Learning vs Continual Fine-tuning. Existing CSSL approaches (a) wait until the end of the continual process to fine-tune, while our proposal (b) leverages knowledge distillation across continual learning steps for both the feature extractor and the classifier.

2. **Evaluation setup reflecting the real world.** We adopted an evaluation setup with a range of evaluation metrics that closely reflect how models perform in the real world, where there is a focus on the performance of the model over the entire continual learning process, instead of only the final model.

3. **Extensive empirical analysis.** We extensively evaluated the proposed training pipeline in various settings against state-of-the-art self-supervised learning techniques, as well as explored the trade-off between different learning objectives. We show that it is robust to catastrophic forgetting in classification tasks while maintaining the quality of the feature extractor.

5.3.1 Approach

In order to continually learn from both unlabelled and labelled data while maintaining a functional classifier, our proposed framework (as illustrated in Figure 5.6) is designed to balance different objectives including knowledge retention, self-supervised learning from unlabelled data, and supervised learning for classification. It consists of two main components: knowledge distillation (KD) and current task learning (CT), one of each for the feature extractor and for the classifier.

Current task learning

This component trains the feature extractor and the classifier using data for the current task (see the right half of Figure 5.6). In order to allow the model to learn from unlabelled data, the feature extractor is trained on the current task using self-supervised learning methods. This component follows the conventional setup of contrastive and Siamese learning approaches, as discussed in Section 2.2.3, and is described in greater detail here: stochastic augmentation functions are first applied to the input which produces two different but correlated views of the input. In this work, we apply random 3D rotation, random scaling and time warping, which were proposed in previous works [Um et al., 2017, Saeed et al., 2019], each with a 50% chance. One of these transformed data samples is passed to the Current Feature Extractor f_t^O, while the other is passed to the Momentum Feature Extractor f_t^T (or the same Current Feature Extractor depending on the self-supervised learning method), to obtain two different embeddings. The embedding from the Current Feature Extractor is passed to an additional, shallow neural network called the predictor h^T. The discrepancy between the embedding from the Momentum Feature Extractor and that from the predictor will then be reduced using the corresponding self-supervised loss function $\mathcal{L}_{\text{FE}}^{\text{CT}}$. The use of an additional predictor here is to allow the feature extractor to be flexible so that it does not have to produce the same embedding for the two augmented views, but should contain the same amount of information.

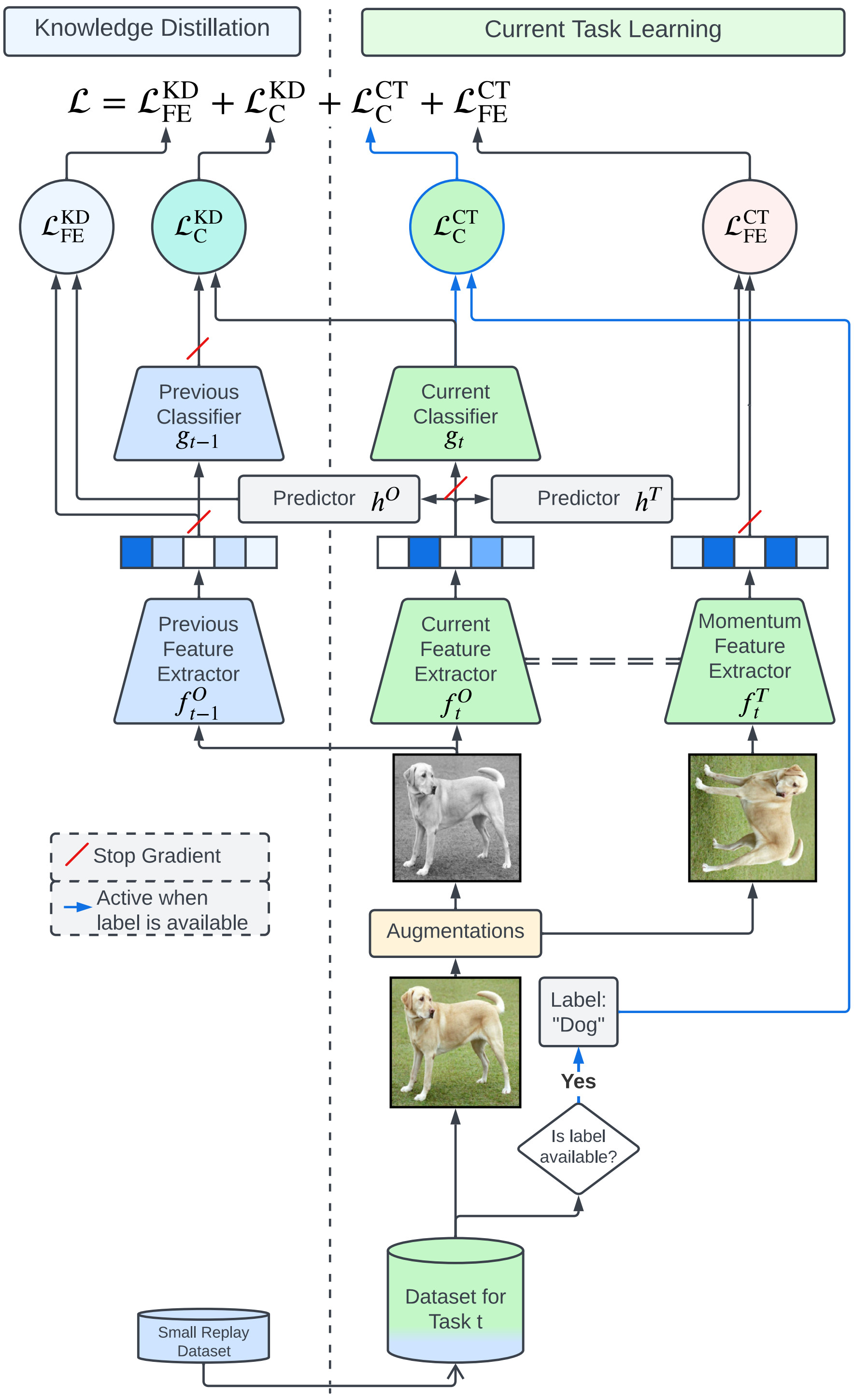


Figure 5.6: Overview of our training framework. Our training framework balances knowledge distillation and current-task learning in an end-to-end manner through a joint loss function. Available SSL methods can be used in training the feature extractors alongside knowledge distillation, while the classifiers are trained on both unlabelled and labelled data through knowledge distillation and fine-tuning.

If the label for the input sample is available, we train the classifier in a supervised manner. The embedding obtained from the Current Feature Extractor will be fed to the classifier network g_t to obtain class probabilities after the softmax activation. Categorical cross-entropy loss $\mathcal{L}_{\mathrm{C}}^{\mathrm{CT}}$ is used for training the classifier. Note that the gradient updates do not back-propagate to the feature extractor. This is to allow the feature extractor to focus on extracting distinctive features from self-supervision, instead of specialising in the current classification task. This is important to ensure that the feature extractor remains general throughout the continual learning process, where the class distribution might shift from one task to another.

Knowledge distillation

Apart from the base learning stage, the model is required to retain knowledge learned from previous tasks while learning from the new data. To achieve this, we make a frozen copy of the trained model (f_{t-1}^{O} and g_{t-1}) from the previous task for knowledge distillation before we start training (see left half of Figure 5.6). For the feature extractor, we adopt a scheme similar to the previous state-of-the-art method, CaSSLe [Fini et al., 2022], where we mirror what we have done for the current task learning, by using an SSL method but with the Momentum Feature Extractor replaced by the feature extractor from the previous step f_{t-1}^{O}. The augmented sample that was passed through the Current Feature Extractor f_t^O will also be fed to the Previous Feature Extractor. Similarly, the discrepancy between the embeddings from the previous and the current feature extractor (after passing through a different predictor h^O) is reduced using the corresponding self-supervised loss function $\mathcal{L}^{\mathrm{SSL}}$. The Previous Feature Extractor is frozen during training and gradients updates will not be applied to it.

Similar to the feature extractor, knowledge distillation is also performed for the classifier using the self-training pipeline (as presented in Section 2.2.4): the predictions from the Current Classifier g_t are made to be similar to the predictions (or soft labels) from the Previous Classifier (g_{t-1}) using the categorical cross-entropy loss. Note that unlike in current task learning, knowledge distillation for the classifier is active regardless of the presence of a label or not for the input sample.

Memory replay

One additional mechanism that our method employs, is the memory replay mechanism in exemplar-based continual learning methods [Rebuffi et al., 2017, Isele and Cosgun, 2018, Rolnick et al., 2019, Mittal et al., 2021], which has been similarly adopted in our previous work (see Section 5.2): a small subset of actual samples or representative samples from previous tasks are kept and replayed during training. This has been shown to be crucial in maintaining model performance and combating catastrophic forgetting in class-incremental learning settings [De Lange et al., 2021].

Overall framework

The mechanisms described above are combined to form the overall loss function (Equation 5.2), which consists of four components: the loss for self-supervised learning $\mathcal{L}_{\text{FE}}^{\text{CT}}$, the loss for knowledge distillation of the feature extractor $\mathcal{L}_{\text{FE}}^{\text{KD}}$, the cross-entropy loss of the classification task $\mathcal{L}_{\text{C}}^{\text{CT}}$, and the loss for knowledge distillation of the classifier $\mathcal{L}_{\text{C}}^{\text{KD}}$.

$$
\begin{aligned}
\mathcal{L}_{\text{FE}}^{\text{CT}} &= \mathcal{L}_{\text{SSL}}\big(f_t^T(x_2)\big), h^T\big(f_t^O(x_1)\big)\big) \\
\mathcal{L}_{\text{FE}}^{\text{KD}} &= \mathcal{L}_{\text{SSL}}\big(f_{t-1}^O(x_1), h^O\big(f_t^O(x_1)\big)\big) \\
\mathcal{L}_{\text{C}}^{\text{CT}} &= \mathcal{L}_{\text{CE}}\big(g_t\big(f_t^O(x_1)\big), y\big) \\
\mathcal{L}_{\text{C}}^{\text{KD}} &= \mathcal{L}_{\text{CE}}\big(g_{t-1}\big(f_{t-1}^O(x_1)\big), g_t\big(f_t^O(x_1)\big)\big)
\end{aligned}
\tag{5.1}
$$

Here the $\mathcal{L}_{\text{CE}}$ refers to the categorical cross-entropy loss, and $\mathcal{L}_{\text{SSL}}$ refers to the self-supervised loss for the particular self-supervised learning method. These losses balance different learning objectives to ensure that the model, including the feature extractor and the classifier, is able to learn from new data while retaining knowledge from previous tasks. This is combined with importance coefficients $\lambda_{\text{FE}}, \lambda_{\text{C}}$ for the feature extractor and the classifier respectively which adjust the importance of different objectives, to form the overall loss:

$$
\mathcal{L}_{\text{Ours}} = (\mathcal{L}_{\text{FE}}^{\text{CT}} + \lambda_{\text{FE}}\mathcal{L}_{\text{FE}}^{\text{KD}}) + (\mathcal{L}_{\text{C}}^{\text{CT}} + \lambda_{\text{C}}\mathcal{L}_{\text{C}}^{\text{KD}}) \tag{5.2}
$$

5.3.2 Experimental protocols

In order to perform a comprehensive evaluation of the continual learning framework proposed in this work, we adopted a similar set of evaluation protocols, as introduced in Section 5.2.2.

Evaluation metrics

For the evaluation metrics, we are adopting the same framework as introduced in Section 5.2.2: using **Final Accuracy (FA)**, **Continual Accuracy (CA)**, **Final F_1 (FF)**, **Continual F_1 (CF)**, **Forgetting (F)**, and **Forward Transfer (FT)** as metrics. This set of values can better reflect the performance of continual learning methods in different use cases.

Experimental setup

Dataset. Similarly, we evaluated our method and baselines using the WISDM2019 [Weiss, 2019] dataset, as introduced in Section 5.2.2. A similar processing pipeline is adopted, with minor modification in the separation of classes: 18 activities classes are evenly separated into 6 tasks with 3 activities each for the continual learning setup. This is to allow longer continual learning scenarios, which is the focus of the work proposed in this section.
Self-supervised learning methods. Since we do not modify the self-supervised learning component from its original formulation [Chen et al., 2020a, Chen et al., 2020b, Grill et al., 2020, Bardes et al., 2022] and the CaSSLe adaptation [Fini et al., 2022], our method is compatible with many existing self-supervised learning methods. We selected a contrastive method, MoCoV2+ [Chen et al., 2020b, He et al., 2020], and an asymmetric-model-based method, BYOL [Grill et al., 2020], as the backbone self-supervised learning method for training. These two methods have been shown to be well-performing in continual self-supervised learning settings [Fini et al., 2022], and our goal is to investigate whether our proposed architecture generalises across different self-supervised learning methods and identify limitations.
Model and hyperparameters. The implementation was built upon the PyTorch [Paszke et al., 2019] implementation by [Fini et al., 2022]. We adopted the same TPN model architecture as used in previous works (see Section 3.3.3, Section 4.3.2, and Section 5.2.2). The batch size is set to 32, and we keep the amount of data replayed to the model during continual learning (for our method) and during classifier training (for other baselines) to be 1% of the original data as this offered the best trade-off between accuracy and amount of labelled data used. The importance coefficients $\lambda_{\text{FE}}, \lambda_{\text{C}}$ for the feature extractor and the classifier are set to 1.0 unless otherwise specified. We keep other hyperparameters unmodified from the previous work by [Fini et al., 2022].
Baselines. We compare our framework against the state-of-the-art continual self-supervised learning pipeline, CaSSLe [Fini et al., 2022], and the *No distill* setup, where no extra measures are taken to mitigate catastrophic forgetting, and the entire model is fine-tuned from task to task. Since the baseline methods are pure self-supervised pre-training methods, the classifier is trained after the feature extractor is fully trained using the task data. These methods can be seen as ablation studies of our proposed framework, since they contain subsets of loss terms used in our proposed method (see Equation 5.2 and Equation 5.3):

$$\begin{aligned}\mathcal{L}_{\text{CaSSLe}} &= \mathcal{L}_{\text{FE}}^{\text{CT}} + \mathcal{L}_{\text{FE}}^{\text{KD}} \\ \mathcal{L}_{\text{NoDistill}} &= \mathcal{L}_{\text{FE}}^{\text{CT}} + \mathcal{L}_{\text{C}}^{\text{CT}}\end{aligned} \tag{5.3}$$

To ensure a fair comparison, these classifiers are trained with memory replay enabled: the same subset of data that our model is trained on, is also available for these classifiers.

We did not compare the work proposed in this section to the one in Section 5.2 because they focus on different aspects of the continual learning process: the current work focuses

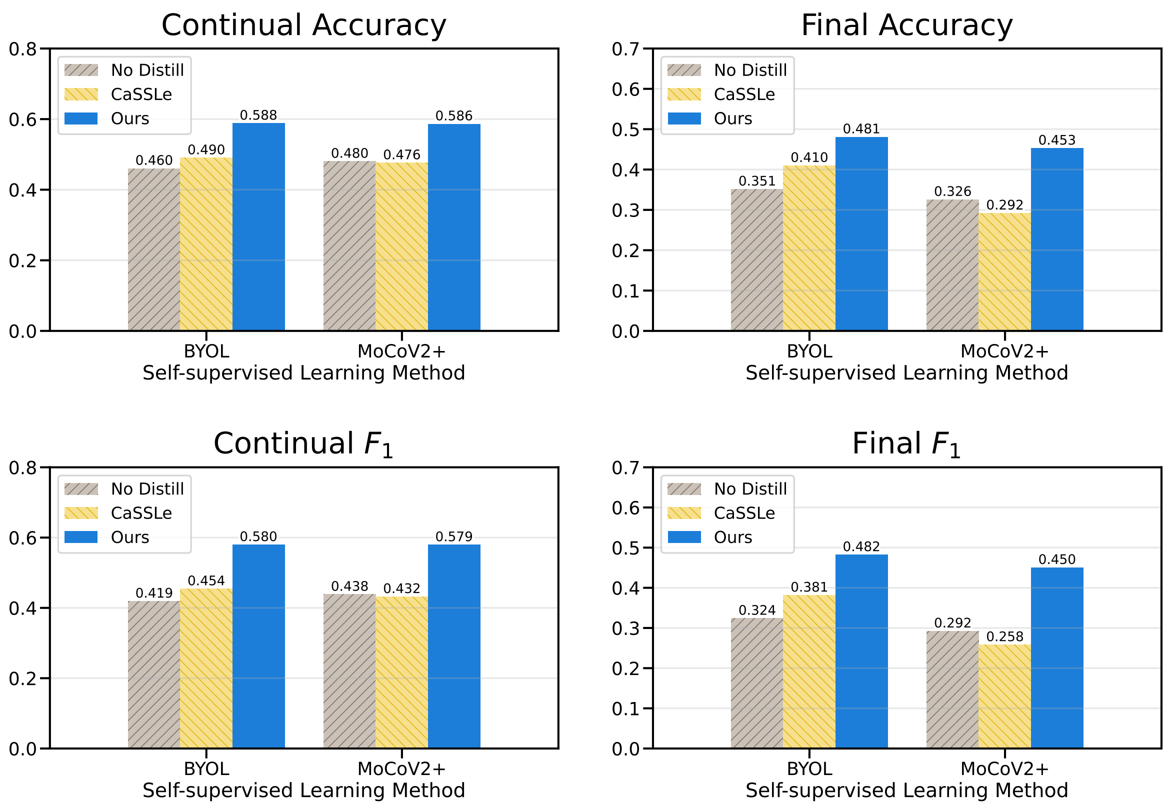


Figure 5.7: Performance comparison between different training methods in terms of Continual Accuracy (upper left), Final Accuracy (upper right), Continual F_1 (lower left), and Final F_1 (lower right). Models are trained using different self-supervised learning methods and knowledge distillation strategies on class-incremental WISDM2019. The figures on the left show the average performance across the entire continual learning process, while the figures on the right show the performance in the final evaluation.

on using self-supervised learning for knowledge retention during continual learning, while the previous work focuses on training a more generalised model at the base training stage.

5.3.3 Evaluation and discussion

Performance comparison against CSSL

As discussed in Section 5.3.2, we focus on the comparison of our proposal against CaSSLe and the *No distill* setup. Figure 5.7 compares the Continual Accuracy, Final Accuracy, Continual F_1, and Final F_1 of these methods with different SSL methods for the feature extractor, on the WISDM2019 dataset split into 6 tasks of equal sizes. Our proposal outperforms the other two baselines irrespective of the evaluation metric or the SSL method, achieving the highest continual accuracy at 0.588, final accuracy at 0.481, continual F_1 at

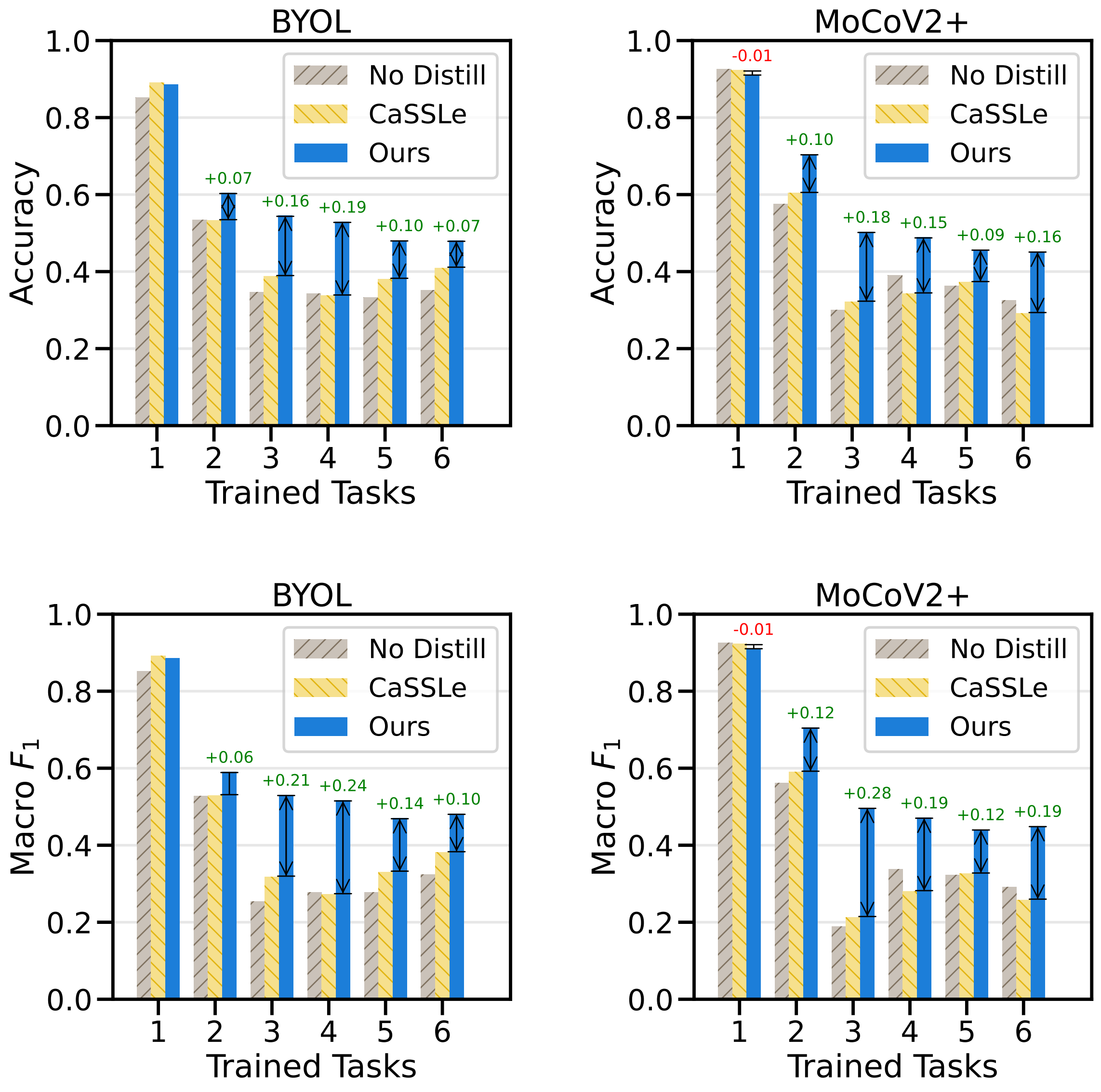


Figure 5.8: Average performance over tasks on WISDM2019. Comparison is drawn between our proposal and baselines using different SSL algorithms across different tasks. Our model consistently outperforms baselines and is robust to forgetting in later tasks.

0.580, and final F_1 at 0.482 using BYOL, outperforming CaSSLe by 0.098, 0.071, 0.126, and 0.101 respectively. It is important to note that the data availability is kept the same across all methods, where at each task, every method has access to data of the current task and 1% of replay data from previous tasks. Our method outperforms the rest by incorporating knowledge distillation and fine-tuning into the pipeline, instead of training the classifier in the end. This validates our hypothesis and shows that our proposal is overall effective in retaining knowledge from previous tasks, by performing well at both the final step and throughout the continual learning process.

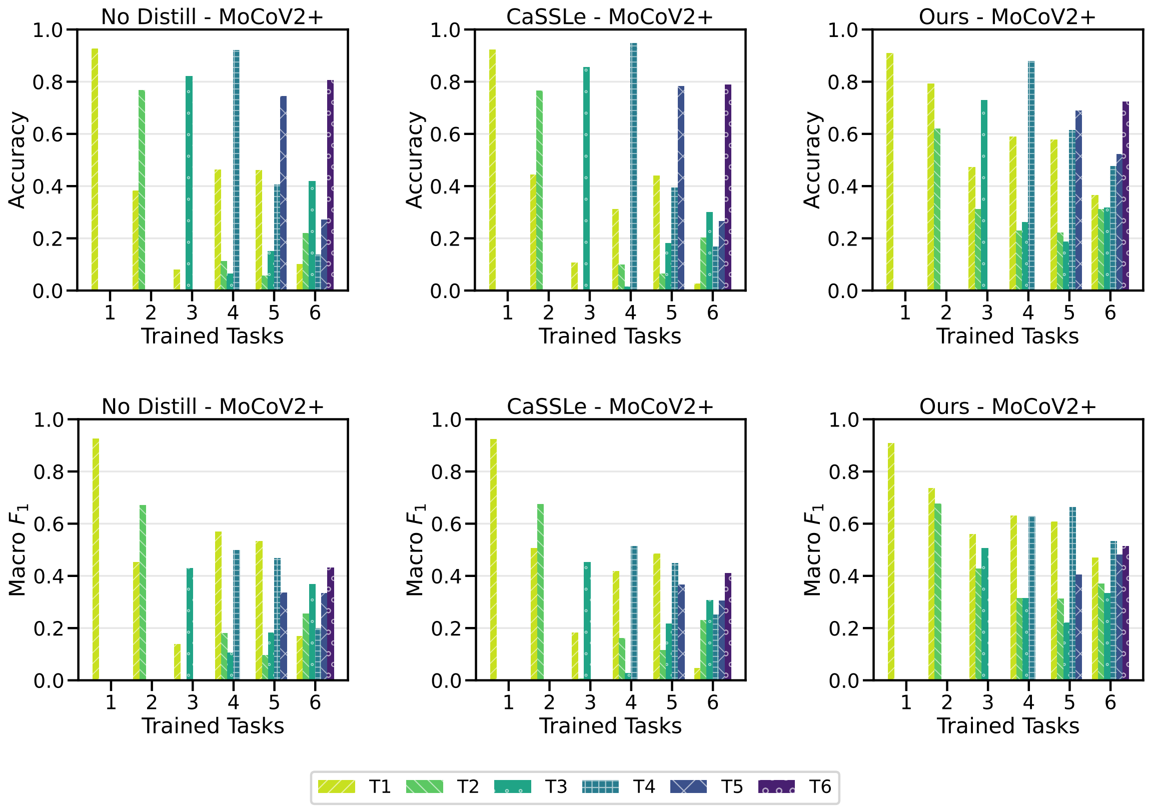


Figure 5.9: Detailed breakdown of performance over tasks on WISDM2019 when using MoCoV2+ as the self-supervised learning method. Fine-grained accuracy and macro F_1 for every task is shown.

Performance variation across time

Here we examine the performance of different methods at different stages of the continual learning process. Figure 5.8 shows the accuracy and macro F_1 over trained tasks of our proposal and the baselines after training on each task. We find that in general, all methods show lower performance after training on more tasks, partially due to the fact that the classification problem becomes more difficult as the number of classes increases, as well as catastrophic forgetting. Echoing the results of the previous evaluation, our method maintains a higher level of performance overall, even though all methods start from similar performance on the first task.

Per-task performance breakdown

To support the intuition that our method prioritises knowledge retention over learning from new tasks, we investigate the per-task performance at each step. Figure 5.9 and Figure 5.10 show the performance of the models on each individual task throughout the continual learning process using MoCoV2+ and BYOL as the SSL method. We find

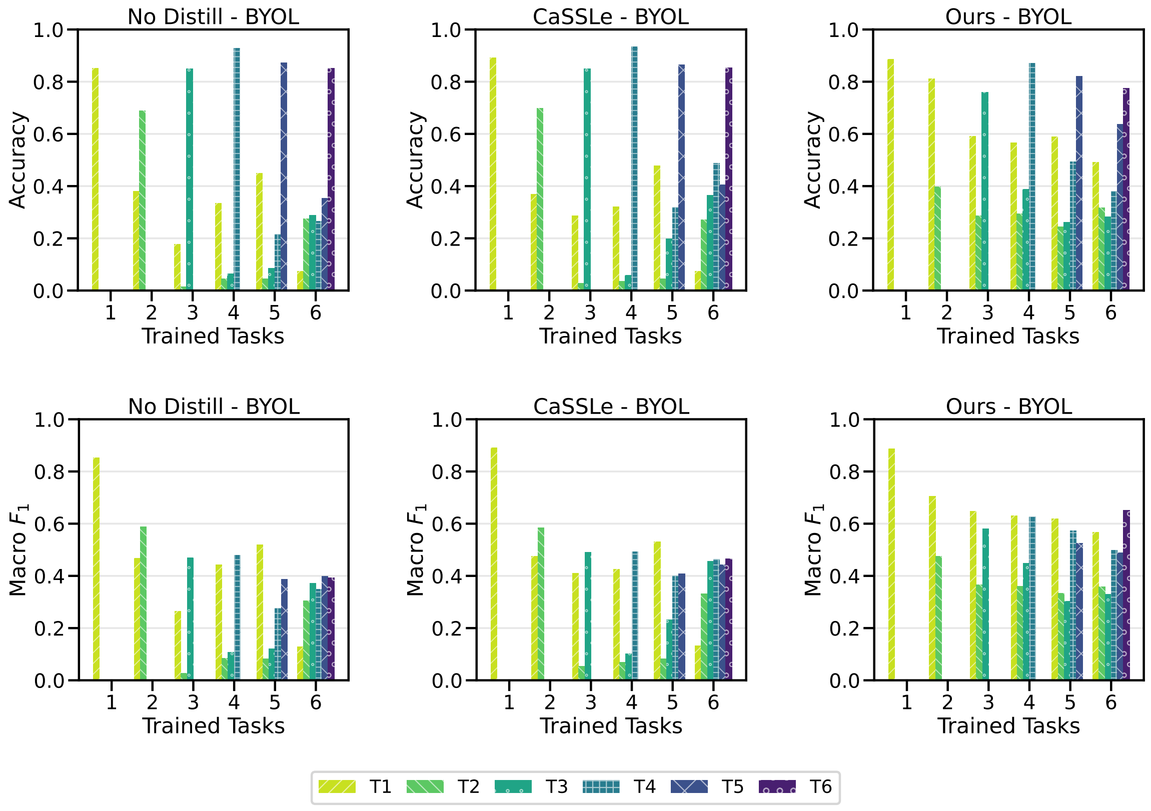


Figure 5.10: Detailed breakdown of performance over tasks on WISDM2019 when using BYOL as the self-supervised learning method. Fine-grained accuracy and macro F_1 for every task is shown.

that our proposal is able to forget acquired knowledge more gracefully [De Lange et al., 2021] than CaSSLe, where the accuracy on previous tasks gradually degrades over time. Without a knowledge distillation scheme for the classifier, CaSSLe, on the other hand, is able to perform the new task better than our framework in terms of accuracy, but the accuracy on previous tasks drops significantly in just one step. When we look at the macro F_1 scores, the advantage of CaSSLe performing well on new tasks, which was displayed in terms of accuracy, is greatly diminished, where the macro F_1 scores remain low for new tasks. This is because macro F_1 scores take into account both precision and recall of all classes. Since CaSSLe and the *No distill* setup perform poorly on previous tasks, and many samples are misclassified as the new classes, this lowers the precision on these classes, and therefore these methods demonstrate lower F_1 overall. On the other hand, our proposal is less affected by this by striking a balance between performance on new tasks and knowledge retention.

Table 5.4: Performance comparisons among different SSL and continual learning methods. We evaluate our proposed framework compared to two continual learning baselines across six metrics on WISDM2019. Our model with BYOL as the SSL method outperforms in most metrics. The best performance for a SSL method is bolded, and that across SSL methods is underlined. An upward pointing arrow indicates higher is better, while a downward pointing arrow indicates lower is better (FA: Final Accuracy, CA: Continual Accuracy, FF: Final F_1, CF: Continual F_1, F: Forgetting, FT: Forward Transfer).

SSL	**Method**	**FA ↑**	**CA ↑**	**FF ↑**	**CF ↑**	**F ↓**	**FT ↑**
BYOL	No Distill	0.351	0.460	0.324	0.419	0.587	0.006
	CaSSLe	0.410	0.490	0.381	0.454	0.527	**0.008**
	Ours	<u>**0.481**</u>	<u>**0.588**</u>	<u>**0.482**</u>	<u>**0.580**</u>	<u>**0.408**</u>	-0.107
MoCoV2+	No Distill	0.326	0.480	0.292	0.438	0.606	0.007
	CaSSLe	0.292	0.476	0.258	0.432	0.662	<u>**0.023**</u>
	Ours	**0.453**	**0.586**	**0.450**	**0.579**	**0.401**	-0.077

Comprehensive evaluation of continual learning and SSL methods

Table 5.4 presents the performance of the proposed framework compared to other baselines on different metrics. These results show that the proposed framework improves the performance across SSL methods as represented by the final and continual accuracy, as well as the F_1 scores. As expected, the absence of distillation in *No Distill* hurts the accuracy and F_1 scores of the overall continual learning framework as it is unable to maintain knowledge with each additional task. This can also be seen in the Forgetting metric where *No Distill* is 0.179 higher than ours and 0.06 higher than CaSSLe when BYOL is used as the SSL method. Our proposal always outperforms CaSSLe across the FA, CA, FF, CF, and F metrics, and achieves the best performance with BYOL. On the other hand, as shown in the results above, CaSSLe and *No Distill* show a slight positive forward transfer in some cases, while our model suffers from a negative forward transfer. This can be explained by the tendencies of CaSSLe and *No Distill* prioritising learning from new data and thus being able to outperform specialised models on new tasks.

Influence of the importance coefficient for classifier training

Since the importance coefficient λ_C (as introduced in Section 5.3.1) specifies the relative importance of the knowledge distillation task compared to learning from new data, it can have a direct impact on the performance of the classifier across time: a higher importance coefficient forces the model to focus on knowledge retention, while a lower coefficient shifts this focus to new task learning.

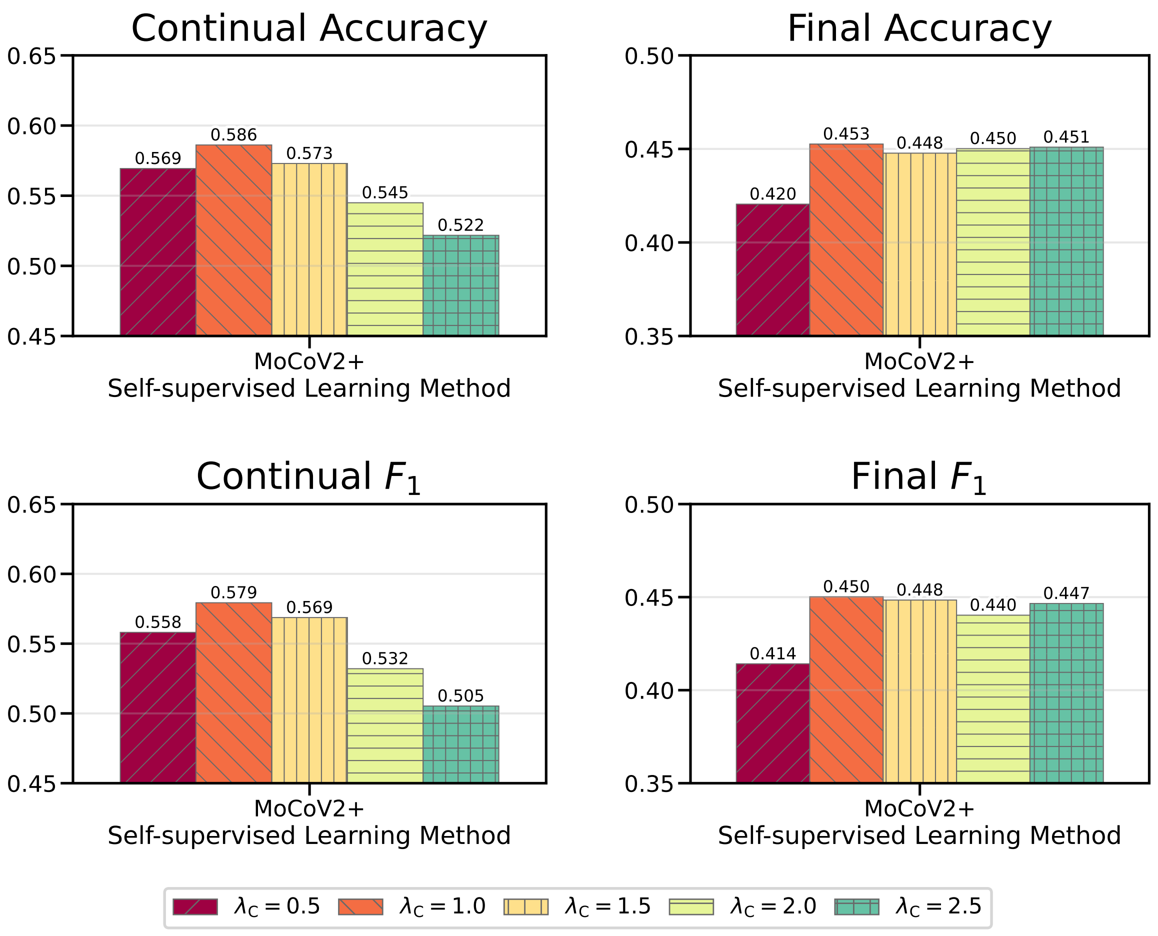
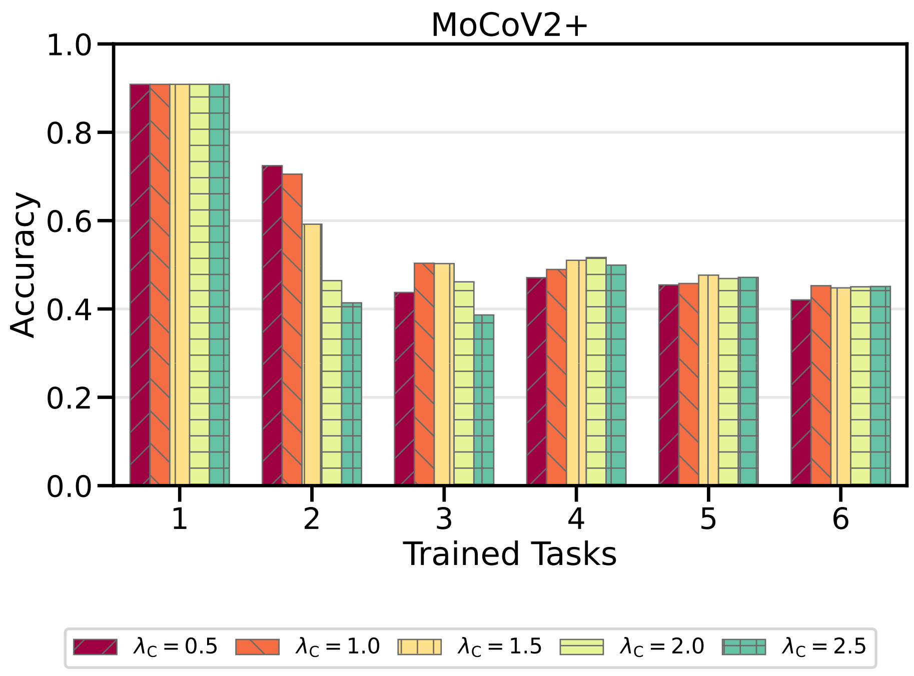


Figure 5.11: Effects of constant importance coefficients. The plot compares the aggregate performance metrics of models trained with the importance coefficient (λ_C) for the classifier set to different constant values.

We conducted an additional set of experiments exploring the effect of this hyperparameter on the performance of the model across time while all other hyperparameters remain unchanged. Figure 5.11, Figure 5.12, Figure 5.13, and Figure 5.14 illustrate the changes in performance when we set the value of λ_C to 0.5, 1.0, 1.5, 2.0, and 2.5. From Figure 5.13, we can see that with lower values of λ_C, the model tends to have higher accuracy in new tasks right after they have been trained on them, which is similar to that of CaSSLe. Higher values of λ_C, on the other hand, force the model to retain knowledge on older tasks, which matches our understanding of this hyperparameter. The macro F_1 scores shown in Figure 5.14 demonstrate trends closer to the overall performance shown in Figure 5.11, where specialising in any particular tasks is penalised as both precision and recall are taken into account.

Furthermore, we observe an interesting trend here: even though higher values of λ_C forces the model to retain more knowledge (see Figure 5.13, where the performance on



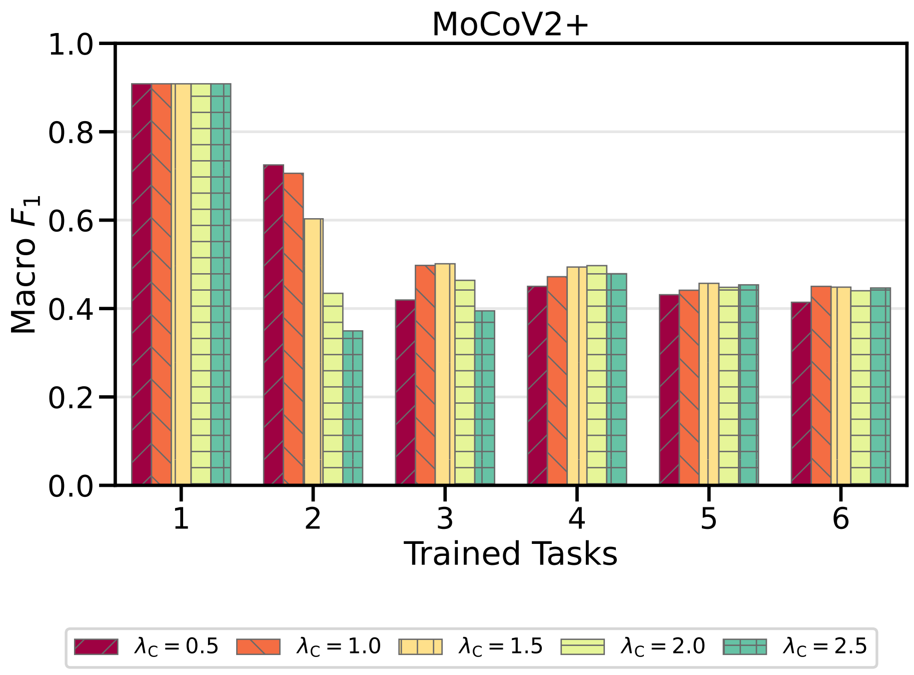


Figure 5.12: Performance after training on different tasks with varying constant importance coefficients (λ_C).

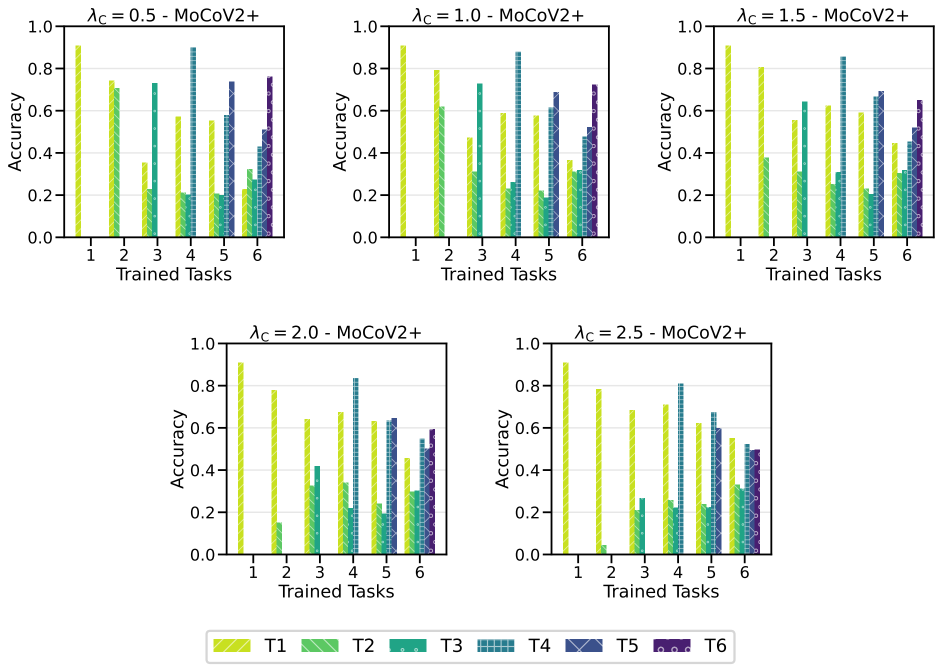


Figure 5.13: Detailed breakdown of accuracy over tasks with constant importance coefficients (λ_C). Fine-grained accuracy is shown for every additional task with different importance coefficients.

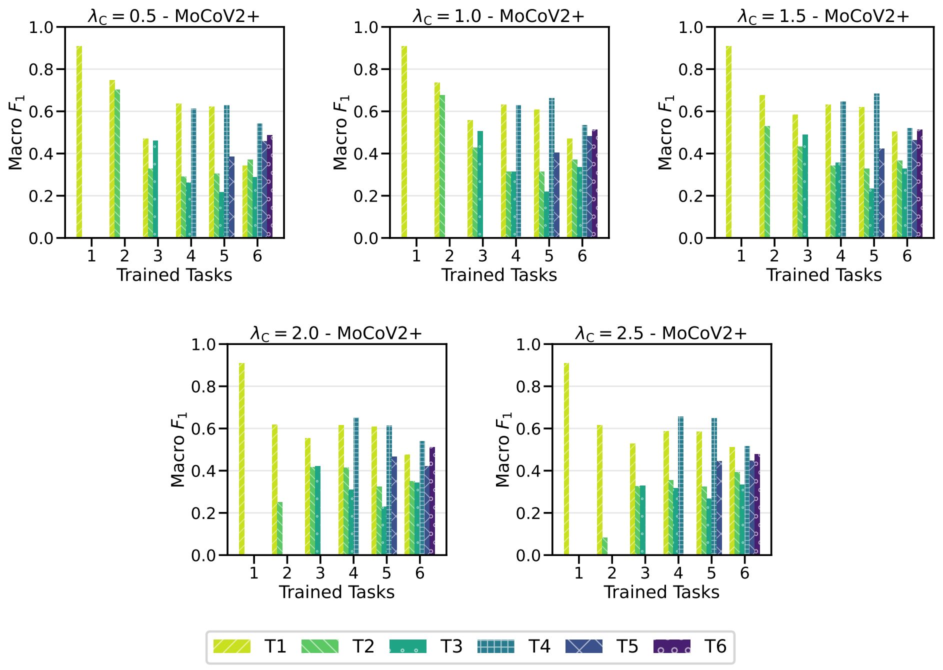


Figure 5.14: Detailed breakdown of macro F_1 scores over tasks with constant importance coefficients (λ_C). Fine-grained macro F_1 score is shown for every additional task with different importance coefficients.

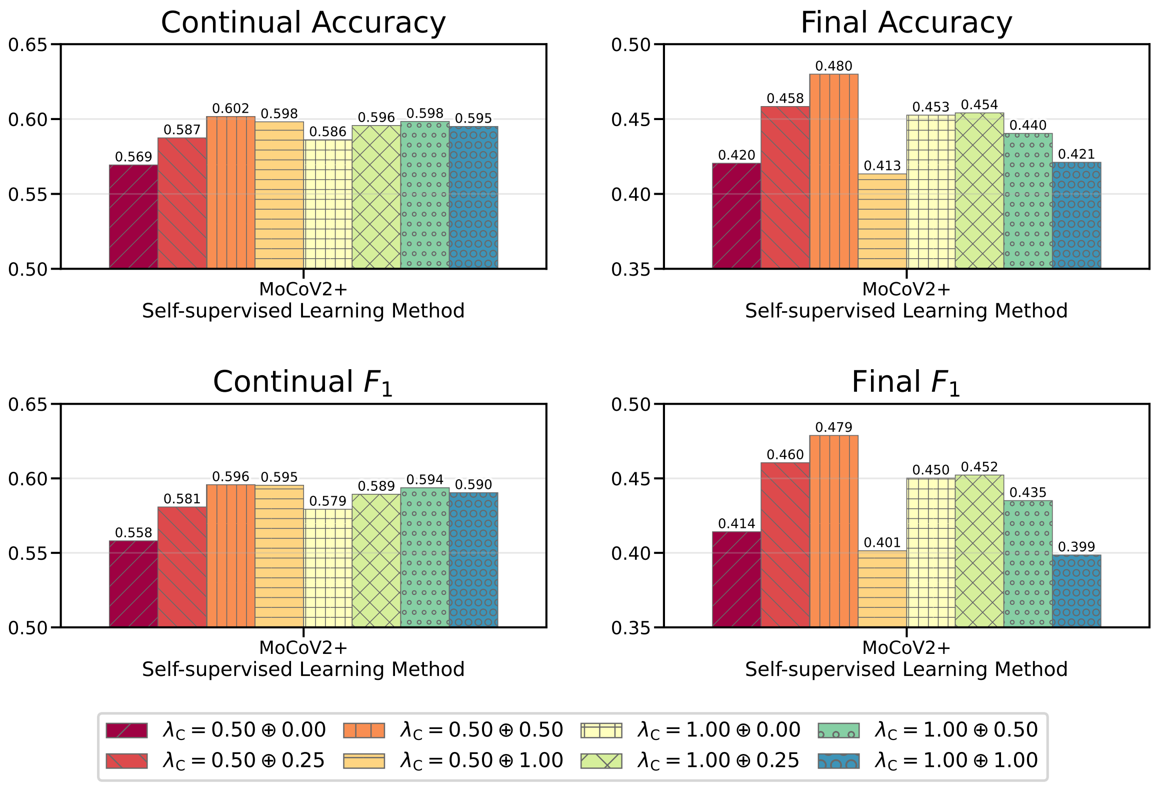


Figure 5.15: Effects of progressive importance coefficients. The plot compares the aggregate performance metrics of models trained with the importance coefficient (λ_C) for the classifier set to different progressive values.

task 1, denoted by T1, remains high throughout all steps for $\lambda_C = 2.5$), the overall performance of the model ends up lower in the long run (see Figure 5.11). This can be counter-intuitive, but explainable by inspecting the role of λ_C: a high value of λ_C sacrifices new task learning for knowledge retention, and this effect *compounds* over time. Performance on the initial task (task 1) is kept at the highest priority when using a high λ_C, while all other tasks suffer as the model is being trained. This is reflected in Figure 5.13, where the performance on tasks 2, 3, 4, 5, and 6 starts off with a much lower value when using $\lambda_C = 2.5$ compared to $\lambda_C = 0.5$. Since the goal of continual learning is to maintain high performance on *all* trained tasks over time, such a trade-off can hurt the overall performance.

We hypothesise that a *progressive* importance coefficient can allow the model to balance knowledge retention and new task learning better than a constant value: as the model gets trained on more classes and tasks, the importance of knowledge retention should increase, and it should be proportional to the number of classes already learned compared to the number of new classes to be learned. To test this hypothesis, we designed a set of experiments where the importance coefficient is increased by a constant value after each task. We denote the setting where λ_{C} is set to a initially, and increased by b after each task as $\lambda_{\mathrm{C}} = a \oplus b$. That is, the importance coefficient at task t is given by

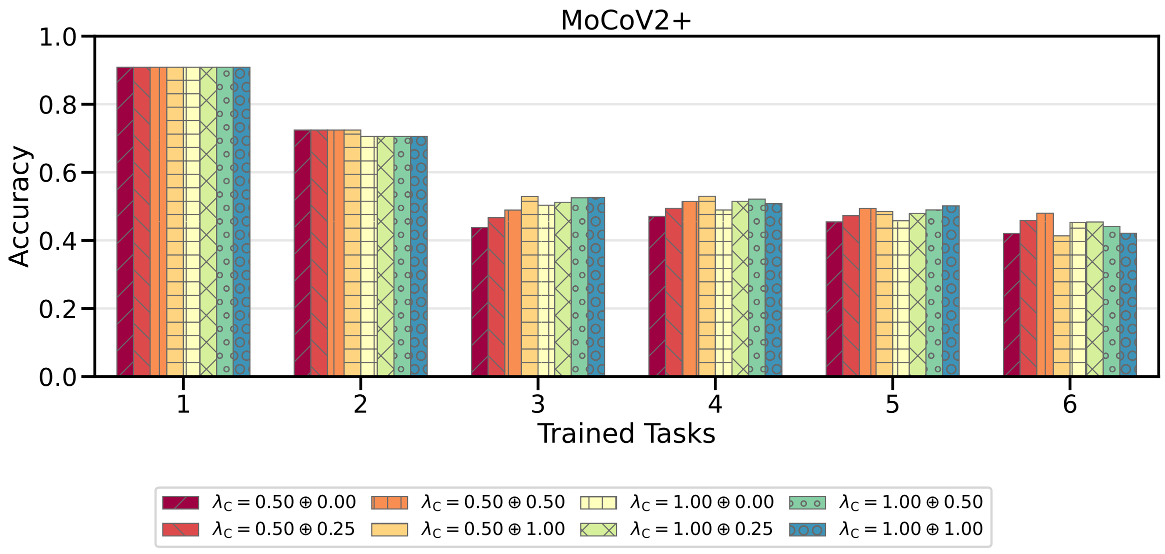
$$\lambda_{\mathrm{C}}(t) = a + b \times (t - 1) \tag{5.4}$$

For example, the setting $\lambda_{\mathrm{C}} = 1.0 \oplus 0.5$ will see values of λ_{C} being set to 1.0, 1.5, 2.0, 2.5, 3.0, and 3.5 for tasks 1, 2, 3, 4, 5, and 6 respectively, while the setting $\lambda_{\mathrm{C}} = 1.0 \oplus 0.0$ is identical to that of a constant $\lambda_{\mathrm{C}} = 1.0$.

We conducted 8 additional experiments with different progressive importance coefficients: $\lambda_{\mathrm{C}} = 0.50 \oplus 0.00$, $0.50 \oplus 0.25$, $0.50 \oplus 0.50$, $0.50 \oplus 1.00$, $1.00 \oplus 0.00$, $1.00 \oplus 0.25$, $1.00 \oplus 0.50$, $1.00 \oplus 1.00$, and the results are shown in Figure 5.15, Figure 5.16, Figure 5.17 and Figure 5.18.

From the detailed breakdown (see Figure 5.17 and Figure 5.18), we can verify our understanding: the performance on trained tasks remains almost unchanged at the last few tasks when the progressive factor is high (see the plots of $0.50 \oplus 1.00$ and $1.00 \oplus 1.00$). The use of a progressive factor allows the model to shift the focus from new task learning to knowledge retention over time. The results shown in Figure 5.15 further support our hypothesis on proportional importance: with $\lambda_{\mathrm{C}} = 0.50 \oplus 0.50$, which corresponds to scaling the importance of knowledge retention proportional to the number of tasks learned, the model is able to achieve the highest continual performance and final performance. A higher value of $\lambda_{\mathrm{C}} = 1.00 \oplus 1.00$ is still able to maintain high performance, but an increased coefficient for knowledge retention at later steps hurts the performance of the final model. Other settings display a spectrum of performance with varying priorities given to knowledge distillation and current task learning.

This set of results allows us to better explore the trade-off between knowledge retention and new task learning, which is one of the most important considerations in continual learning.



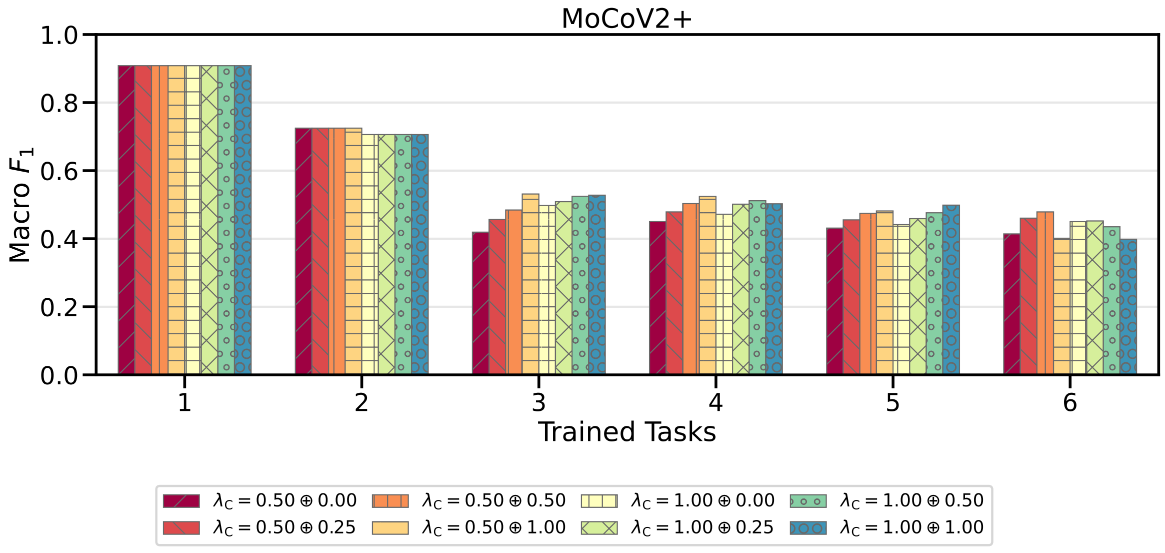


Figure 5.16: Performance after training on different tasks with varying progressive importance coefficients (λ_C).

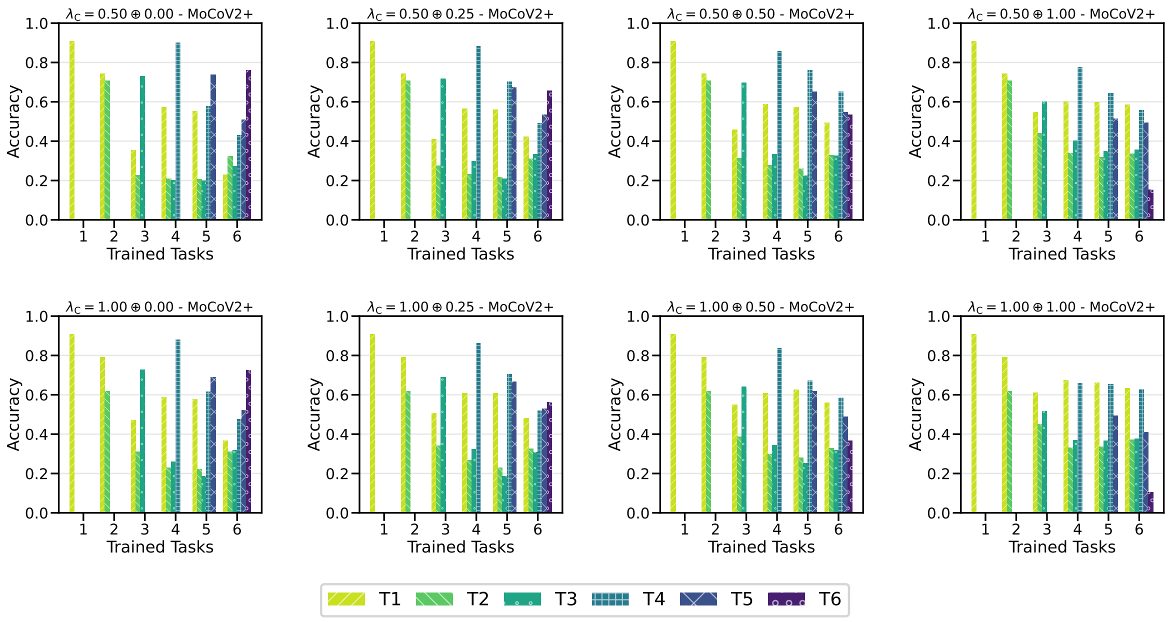


Figure 5.17: Detailed breakdown of accuracy over tasks with progressive importance coefficients (λ_C). Fine-grained accuracy is shown for every additional task with different importance coefficients.

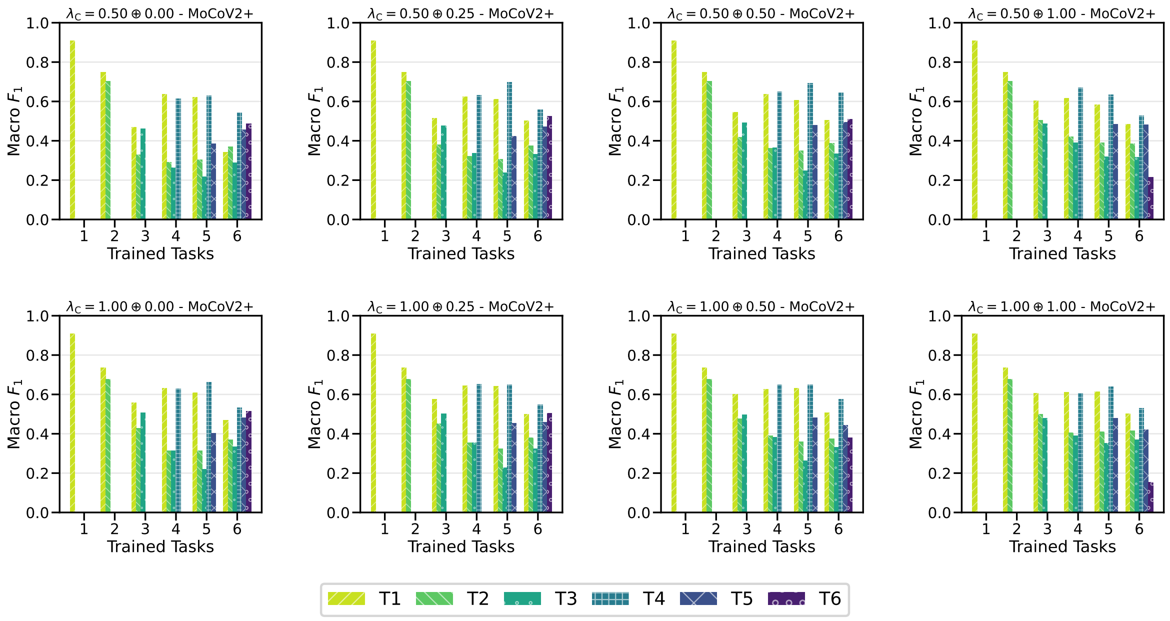


Figure 5.18: Detailed breakdown of macro F_1 scores over tasks with progressive importance coefficients (λ_C). Fine-grained macro F_1 score is shown for every additional task with different importance coefficients.

5.4 Conclusions

Deployment of mobile sensing models in the real world comes with various challenges, one of which is the changes in user behaviour over time. Devices may need to perform new tasks within their lifespan, but deep learning models suffer from catastrophic forgetting when they are trained on new data. In this chapter, following the motivating examples in Section 5.1, we studied continual learning for HAR from two angles – optimising the training algorithm at base model training, and the knowledge retention mechanism during incremental steps.

Section 5.2 demonstrated that modifying the conventional training setup at initial learning can have a positive impact on the following continual learning steps. Our multi-task learning framework guides models to extract more generalisable representations, which improves overall performance. We explored the impact of multi-task configurations, exemplar quantities, and different losses on model performance, and the results demonstrated improved performance in HAR when a diverse of tasks is used, measured in a comprehensive set of continual learning metrics. This work opens the door to improving the quality of the base model in continual learning, which motivates the exploration of further generalisation techniques in the future.

Section 5.3 introduced a continual learning method that addresses the shortcomings of existing self-supervised and supervised continual learning approaches, by enabling end-to-end continual classifier and feature extractor training with the possibility of deployment at any point. We carefully designed the learning objective to balance feature extractor and classifier training, as well as knowledge retention and learning from new data. Through extensive evaluation and a comprehensive set of evaluation metrics, we have demonstrated that the proposed framework is able to balance and explore the trade-off between knowledge retention and learning from new data compared to the state-of-the-art, and achieves higher performance overall.

These works demonstrate the potential of utilising different sources of supervision, including multi-task and self-supervised learning, for developing human activity recognition systems that can adapt to changes in user behaviours while leveraging data efficiently. The ability to continually learn in real-world non-stationary environments will be an important capability as mobile sensing applications evolve.

Chapter 6

Conclusion

In this thesis, we presented novel training frameworks drawing on different training paradigms in deep learning, for the development of data-efficient, flexible and well-performing human activity recognition systems. Our work was motivated by the difficulty in obtaining high-quality, labelled data in mobile sensing, and it is predicated on the premise that training paradigms including self-supervised learning, self-training, multi-task learning, and contrastive learning, combined with abundant unlabelled data, are the key ingredients in overcoming these limitations. In this concluding chapter, we revisit the key contributions and limitations of our work, and suggest potential research directions beyond this thesis.

6.1 Summary of contributions

6.1.1 Improving human activity recognition through self-training with unlabelled data

In Chapter 3, we presented a novel training framework that is able to make use of large unlabelled datasets collected in free-living environments for human activity recognition. Motivated by the success of prior works, our proposal leverages the advantages of teacher-student self-training and self-supervised learning to form a training framework that can effectively increase the diversity of data for the models to be trained on. Our proposal is able to boost the performance of HAR models using the same amount of labelled data and without increasing the complexity of the models, when compared to previous state-of-the-art training frameworks. We further demonstrated that our models are able to perform well when the availability of labelled data is low. These findings contribute to the development of data-efficient HAR systems in the area of ubiquitous computing, and effectively address the constraints posed by labelled datasets.

6.1.2 Collaborative contrastive learning for human activity recognition

In Chapter 4, motivated by the increasing popularity of multi-device setups, we studied the problem setting of time-synchronous multi-device systems. Based on the observation that time-synchronised data from multiple devices are capturing the same physical activity from different angles, and can be seen as natural transformations of each other, we developed a contrastive learning framework to train an effective feature extractor without relying on hand-crafted transformation functions. This contrastive learning task involves clustering time-aligned data, while pushing misaligned samples farther apart in the embedding space, enabling us to extract supervisory signals from unlabelled multi-device datasets. We designed a specialised sampling algorithm and a custom loss function tailored to this setting, and we demonstrated that the proposed approach was able to outperform other existing methods in self-supervised learning. Additionally, we showcased its resilience against common sensor noises. This work expands the range of data sources that can be used to develop data-efficient human activity recognition systems and demonstrates the value of time-aligned data in mobile sensing.

6.1.3 Overcoming catastrophic forgetting with multi-task and self-supervised learning

In Chapter 5, we investigated how different training paradigms can be used to mitigate the performance degradation caused by changes in user behaviour, in the setting of continual learning. We introduced two training pipelines to address this issue from different angles. The first approach examined how multi-task learning can guide deep-learning models to discover more generalisable solutions during the initial training phase for continual learning when a relatively large amount of labelled data is available, by using subsets of the base classes to set up multiple tasks. The results demonstrated that our proposed multi-task learning setup allows models to adapt better to new tasks while reducing performance degradation on previous tasks. The second approach focused on the later continual learning steps and expanded on previous works in the area of continual self-supervised learning. We addressed the impractical assumption of large amounts of labelled data being available at the end of training in existing works by proposing a unified training pipeline that leverages self-supervised learning and self-training to balance the objectives of knowledge retention and new task learning. We demonstrated that our proposal is able to balance these objectives for both the feature extractor and the classifier, and additional experiments showed that it is capable of exploring the trade-off between different continual learning objectives. These works consider a real-world setting in which models need to adapt to new tasks, and they contribute to the development of practical and data-efficient solutions in this setting.

6.2 Implications and limitations

The frameworks and findings presented in this thesis hold promising implications not only for the mobile computing community but also beyond. Our work focused on utilising different sources of supervisory signals for the development of well-performing and data-efficient human activity recognition systems. Researchers can build upon our methods, ideas, problem settings and models to develop systems that are more efficient, personalised, and achieve the goal of bringing mobile sensing to every device. Beyond human activity recognition, since our proposals are generally modality-agnostic, the training algorithms could also benefit other mobile sensing tasks as well as other areas in machine learning, including computer vision and audio processing. Developers and engineers could build software and tools that are context-aware and understand user behaviour by employing our models, and develop solutions that enable better user experience and interaction with various mobile devices. These software and tools also possess utility in healthcare environments, facilitating ongoing monitoring and comprehension of user behaviours. This potential utilisation opens up novel avenues for enhanced and personalised healthcare solutions. Furthermore, understanding each individual's behaviour not only has a direct impact on their lifestyle, but also has a potential for impact on a broader scale, informing public health policies.

In the meantime, it is important for us to identify the limitations in our studies so that the findings can be understood in the correct context, and future research can be conducted to address them. Although our works have been evaluated on various publicly available datasets, these datasets come with their own limitations. They were mostly collected from a small set of individuals coming from specific geographical areas. User behaviours can differ significantly across different demographics and across time, and this means that changes to our proposed frameworks could be needed in order to adapt to other settings. Ensuring fairness is an important aspect of personalised computing, and therefore further studies are needed to verify how well our proposals can achieve this. Large-scale population datasets such as UK Biobank [Sudlow et al., 2015], which contains data representing a wider set of users, can potentially alleviate this problem. However, utilising these large datasets could introduce new challenges in privacy, training efficiency and resource management.

In addition, the main focus of our studies has been on the training pipelines and algorithms. For the deployment of these training pipelines, other practical considerations would need to be taken into account. One of which, is the model architecture for the recognition algorithm. While we have consistently chosen lightweight models for our studies, it is important to consider that the feasibility of deploying these models might vary depending on the computing capabilities of the target devices. In certain cases, these models could either prove impractical to execute or underutilise the available resources on these devices. Different model architectures can be accelerated to varying extents by hardware accelerators [Habib and Qureshi, 2022], implying that each device would have its own optimal model architecture. Our proposed frameworks were able to outperform the previous state-of-the-art frameworks using the TPN model architecture (see Section 3.3.3). However, further studies would be necessary to verify whether this holds across architectures and whether different training methods could be proven to work better with different architectures.

Furthermore, across our work, we have modified the training pipeline to take advantage of more data, training at multiple stages, and learning from additional tasks. These additional training steps naturally imply that more computation is needed for training, which could incur additional overheads and costs. However, this cost is limited to the training step, because the improved performance does not come from a change in the model architecture at runtime. Since model training is usually done centrally, and the models are relatively lightweight, the impact on practical deployment should be limited. However, if any of these assumptions changes, further considerations are needed to select the best training framework for the specific purposes.

Another requirement for our proposed frameworks is the additional sources of data. In particular, unlabelled data (used in Chapter 3 and Chapter 5) and time-synchronised data (used in Chapter 4) are needed in our proposals, and the collection of these data could incur additional costs. However, as we have discussed in Chapter 1, these data can be collected at a much lower cost compared to labelled data because they can be collected passively without requiring much interaction from the user. Our works leverage this unique opportunity in mobile sensing, thereby reducing the reliance on labelled data, which is much harder to collect.

Following up on the previous point, the additional data requirement might lead to privacy concerns. Users might not be willing to transmit data collected passively in the background to researchers or device manufacturers due to privacy reasons, especially those collected from multiple devices simultaneously. Research in on-device user-adaptation [Matsui et al., 2017] and personalisation [Sani et al., 2018, Craighero et al., 2023] for deep learning models could be potentially leveraged to reduce the need for large-scale data collection. Also, federated learning approaches [Shi et al., 2021] could be explored, where anonymised model updates are computed on-device and sent to a central server, instead of requiring raw sensor data to be uploaded.

Our proposals in Chapter 5 address the problem of continual learning, providing useful tools to handle changes in user behaviours. While continual learning is an important problem in the broader machine learning area, there are limitations and practical concerns when applying it to mobile sensing tasks, such as human activity recognition. In real-world applications, it is infeasible to collect sufficient data from all possible human activities. Instead of solely relying on user feedback, models should be able to detect changes in user behaviours, such as when a user performs a new activity that needs to be modelled. Although our proposals for tackling continual learning aim to improve model generalisability and leverage unlabelled data, we have not addressed the issue of activity detection, where models first detect new activities and then trigger further data collection and training. This is related to the open-set nature of activity recognition [Chen et al., 2021], in which a realistic solution should reject unknown activities, especially given that users often perform background or irrelevant activities in day-to-day life. Existing works tackling open-set recognition problems have explored different approaches, such as evidential deep learning [Bao et al., 2021], which is based on uncertainty estimation for rejecting unknown classes, thresholding based on inter-class confidence [Shu et al., 2018], and using fake samples from a Generative Adversarial Network (GAN) for the unknown class [Yang et al., 2019]. However, many of these solutions come from the video-based human activity recognition community. It is worth exploring whether these approaches are readily applicable to sensor-based human activity recognition, for example, in a hybrid approach combining continual learning and activity detection, which initiates automatic continual updates according to changes in user behaviours, addressing one of the critical problems in real-world deployment of mobile sensing models.

Furthermore, our investigations have centred on a specific mobile sensing task, and the potential for generalising these findings to other mobile sensing contexts requires further verification. For instance, the unexplored territory includes investigating whether models designed for activity recognition can be adapted for tasks such as fall detection or long-term user monitoring. Other machine learning tasks, such as object recognition and natural language processing, could potentially leverage some of our proposed training frameworks since our proposals are themselves modality-agnostic. Furthermore, there have been instances of large-scale foundation models that can handle multiple tasks being developed in other domains [Floridi and Chiriatti, 2020, Radford et al., 2021]. It remains to be seen whether mobile sensing can effectively embrace these techniques.

In summary, we recognise that deploying a practical mobile sensing model involves various facets. While our proposed training frameworks do not address every one of them, we are confident that our careful research design effectively addresses the specified research questions within the defined scope of our exploration. Our research focused on modelling user behaviour through sensor time series and offered effective and data-efficient solutions within this domain.

6.3 Future research directions

This thesis has presented progress in the area of ubiquitous computing, but real-world deployment of mobile sensing models remains challenging. In closing, we highlight several promising research avenues within this field that warrant in-depth exploration. These directions are crucial for achieving the overarching objective of enabling device intelligence and personalised computing accessible to all.

6.3.1 Domain generalisation

One of the biggest challenges in the real-world deployment of mobile sensing models is to achieve 'train once, deploy everywhere' [Laskaridis et al., 2021, Qin et al., 2022], where models are generalisable to different wearing positions, devices, users and demographics [Morales and Roggen, 2016, Jiang et al., 2018, Hasthanasombat et al., 2022, Bento et al., 2022, Qin et al., 2022]. While our research has showcased the success of harnessing unlabelled data to enhance the adaptability of deep learning models across users, it is important to note that additional considerations become pertinent when these models are implemented across various wearing positions and devices. Given that there are countless potential variations and settings for mobile sensing, the ideal solution would be to have a global model that works across settings, requiring as little adaptation as possible for deployment.

Unsupervised domain adaptation (UDA) has been studied by many researchers [Ganin and Lempitsky, 2015, Chen et al., 2019, Chang et al., 2020, Wilson et al., 2020, Zhou et al., 2020, Hu et al., 2023] and has been proposed as a promising solution for a model to be adapted to different domains using unlabelled data. Within this area, learning domain-invariant features [Chen et al., 2019, Wilson et al., 2020, Zhou et al., 2020] has been one of the key ingredients in the training schemes proposed, in which the models are trained with data from a wide range of domains, so that it can extract domain-generalisable features. Learning to extract invariant features is also the goal of many other training paradigms, including the use of data augmentation [Kalouris et al., 2019], contrastive learning [Cai et al., 2020], and as we have seen, in domain adaptation. It would be worthwhile to explore how different training paradigms can be combined to train more generalisable models. For example, our ColloSSL framework proposed in Chapter 4 utilised cross-device contrastive learning to learn more robust representations. This could potentially be extended with domain adaptation techniques to train models that can generalise to different devices and sensors.

In addition to only learning domain-invariant features, [Qin et al., 2022] argued that *domain-specific* features can also be fused with them to further boost model generalisability. The Adaptive Feature Fusion for Activity Recognition (AFFAR) framework separated the learning objective into three parts: domain-specific representation learning, domain-invariant representation learning and classification learning. The proposed scheme dedicates parts of the network to capture domain-specific features and calculates the similarity between different domains, which is then used to adaptively fuse features extracted by the domain-specific networks. The authors have demonstrated that this achieves higher model generalisability compared to other methods, including adversarial adaptation and meta-learning. This line of work holds the potential for establishing model generalisability across diverse settings, encompassing devices, sensors, and users. The exploration of how different training paradigms could be harnessed to extract domain-specific features is a promising avenue to investigate.

Going back to the fundamentals, it is worth considering that deep learning models might exhibit sensitivity to various factors in the data generation process for human activity recognition. These factors include environmental conditions and user attributes. In response, recent studies by [Bento et al., 2022, Hasthanasombat et al., 2022, Zhang et al., 2022] have explored more principled designs for neural networks and input representations to address this concern. In particular, instead of using raw sensor signals as input, frequency-based features have been studied [Bento et al., 2022, Hasthanasombat et al., 2022], and they demonstrated better generalisability across domains. This could potentially be explained by the nature of certain activities, including sitting, walking and running, that they are periodic and occupy lower-bands of frequencies compared to that of sensor noises. Integrating domain knowledge into the setup and training of neural networks, as proposed in these works, could prove advantageous. It would be interesting to investigate the use of frequency-domain features for developing more generalisable models.

Overall, I believe that in order to achieve the goal of 'train once, deploy everywhere', a mixture of different training techniques and paradigms is likely to be needed, especially when models are tailored to each particular use case.

6.3.2 Multi-modal sensing and foundation models

In this thesis, we mainly focused on using sensor signals coming from accelerometers and gyroscopes for human activity recognition. Other data sources, including sensors with different modalities, are underutilised in our studies and can potentially be leveraged. Works including *Sense and Learn* [Saeed et al., 2021] and *COCOA* [Deldari et al., 2022], as introduced in Section 2.2, have started looking at cross-modality modelling, where data from multiple modalities are used in conjunction for training. This reflects one of the current topics in the area of machine learning – foundation models, which are trained on a wide range of data modalities and can be adapted to different downstream tasks [Bommasani et al., 2021]. Emergent capabilities and high performance have been observed in several machine learning modalities, including natural language [Floridi and

Chiriatti, 2020, Touvron et al., 2023], vision [Wang et al., 2022] and audio [Chen et al., 2024]. However, mobile sensing is still lagging behind in this regard. Recently, the work by [Girdhar et al., 2023] has proposed a unified embedding scheme, ImageBind, for six data modalities, including images, text, audio, depth, thermal, and inertial measurement unit (IMU) data. The embeddings across different modalities are aligned using the contrastive learning setup with co-occurring data pairs. For the IMU modality, the alignment is performed using a dataset that captures both egocentric camera feeds and IMU data. The potential in this direction of research can be significant, where the development of a foundation model for mobile sensing could be very useful for many downstream tasks, but many research challenges remain, especially in terms of the resource constraints in mobile sensing. Opportunities exist in leveraging pre-trained models from other areas with significantly higher capabilities than mobile sensing, using an alignment scheme similar to IMU2CLIP [Moon et al., 2022] or ImageBind [Girdhar et al., 2023].

Although [Tong et al., 2020] argued that IMUs simply do not provide enough information for performing the activity recognition task, and they argued for the use of imagers in place of IMUs, I am of the opinion that by integrating additional sources of data that are currently underutilised, such as data from multiple devices, location data, environmental sensors, user-device interactions, linguistic patterns, and leveraging advanced multi-modal foundation models, we can craft more robust, generalisable and well-performing recognition algorithms. These algorithms would offer a comprehensive perspective on the user's behaviour, resulting in a more holistic behavioural analysis.

6.3.3 Generative methods

Following the discussions in the previous section, success in other data modalities has been attributed to the use of generative modelling, in which the task is to model the probabilistic distribution of the training data [Bond-Taylor et al., 2021]. In the area of mobile sensing, researchers have attempted to use Generative Adversarial Networks [Goodfellow et al., 2014] to perform generative modelling [Wang et al., 2018, Yoon et al., 2019, Li et al., 2020], as introduced in Section 2.2.2. However, only limited success has been observed, potentially due to large data variations and sensor noises present in sensor data.

A different approach adopted by other researchers in mobile sensing is to generate synthetic data from other data sources [Kwon et al., 2020, Leng et al., 2023] or using statistical conditioning [Zuo et al., 2023]. The work by [Kwon et al., 2020] proposed an interesting framework in which IMU data are 'synthesised' using video data. In this work, the authors leveraged readily available videos from the Internet, depicting individuals engaging in various activities. They performed 3D body tracking on these videos and generated IMU data by virtually placing sensors on the resulting 3D model. Taking one step further, [Leng et al., 2023] proposed the use of large language models and multi-modal generative models to synthesise IMU data from text. The authors first used ChatGPT, a large language model, to generate long and detailed descriptions of a certain activity. These descriptions are then fed to T2M-GPT, a model capable of performing text-to-3D-motion-sequence translation. Similar to the work by [Kwon et al., 2020], these 3D motion sequences are used to generate IMU data by simulating virtual IMUs in the 3D space. In another work, inspired by the success seen in synthesising realistic samples for other modalities, [Zuo et al., 2023] adopted diffusion modelling together with statistical conditioning to generate realistic samples for human activity recognition. A diffusion model typically learns to iteratively denoise data, which can be used to generate data from random noise by iteratively applying this process. The proposed framework utilises mean values, z-scores, and skewness in an unlabelled dataset to generate new synthetic data, which are then used for further training. This line of research offers a new path in obtaining training samples for mobile sensing at a much lower cost compared to direct data collection. It would be interesting to see how these techniques and the multi-modal foundation models discussed in the previous section could be used in conjunction.

6.3.4 Collaborative and adaptive sensing

In addition to data modelling, other practical considerations, such as power efficiency and computing capability, are also necessary for the real-world deployment of mobile sensing models. Collaborative and adaptive sensing, as introduced in Section 2.1.2, is an important research direction in developing energy-efficient and user-friendly systems. We have seen works in Section 2.1.2 dedicated towards sensor and device selection strategies for better utilisation of resources. [Min et al., 2019] looked at how a runtime quality assessment system can be used to select the best sensors and devices for a particular recognition task, and proposed a duty cycle in which not all sensors need to be turned on all the time. An interesting direction of research would be to examine how reinforcement learning, an area of machine learning research that looks into training intelligent agents taking actions in an environment [Sutton and Barto, 2018], can be used in conjunction with different assessment schemes to develop a system which selects the best sensors and devices. [Murad et al., 2020] looked at how information-theory-based metrics can be used for adaptive sensing with IoT devices that are extremely resource-constrained, and proposed a framework using Gaussian processes as the predictor and neural networks as reinforcement learning agents to optimise energy efficiency. It would be worthwhile to explore whether this can be adapted to other sensing tasks such as wearable-based human activity recognition, as well as how models used in adaptive scheduling and those used in activity recognition can be trained concurrently to form a unified mobile sensing system.

6.3.5 Beyond activity recognition

Moving beyond detecting user activity at a particular point in time, research remains to be done on how these personal observations can be used for longitudinal monitoring, healthcare applications, population-scale analysis and policy making. For example, works including those by [Rovini et al., 2017, Lin et al., 2021, Han et al., 2022, Merrill and Althoff, 2023] looked at how diseases can be detected and monitored using mobile sensing models. In the context of the recent pandemic, these can be crucial for informing public health policies, especially when observations are collected at scale. Datasets such as Homekit2020 [Merrill et al., 2023], which contain sensor data and other observations such as symptom reports and PCR influenza test results, can be crucial tools for such purposes. [Mezlini et al., 2022] also looked at using activity data collected at scale to estimate the burden of respiratory diseases on a population. They found statistical significance in changes in behaviour caused by respiratory diseases and captured seasonal behavioural patterns. [Hucklesby and Holt, 2023] have presented how monitoring using wearable technologies can be beneficial in handling various social issues, as well as the risks that come with it.

Mobile sensing should not be limited to only passively observing human behaviour. It should also be capable of actively empowering individuals and society as a whole. While our work has focused on individual activity recognition, the ultimate goal is to develop technologies that can offer advice, suggestions and insights for individuals and beyond. Future research should delve into expanding the capabilities of these technologies, exploring how they can be leveraged to benefit not only individuals but also wider communities. By broadening the scope of current research with care, diligence, and imagination, ubiquitous computing holds the promise of a future where technology facilitates positive change, transforming lives for the better.

Bibliography

[Abadi et al., 2016] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: a system for large-scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283.

[Aggarwal et al., 2008] Aggarwal, P., Syed, Z., Niu, X., and El-Sheimy, N. (2008). A standard testing and calibration procedure for low cost mems inertial sensors and units. *The Journal of Navigation*, 61(2):323–336.

[Ahsan et al., 2019] Ahsan, U., Madhok, R., and Essa, I. (2019). Video jigsaw: Unsupervised learning of spatiotemporal context for video action recognition. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 179–189. IEEE.

[Aljundi et al., 2018] Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., and Tuytelaars, T. (2018). Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 139–154.

[Almaslukh et al., 2017] Almaslukh, B., AlMuhtadi, J., and Artoli, A. (2017). An effective deep autoencoder approach for online smartphone-based human activity recognition. *Int. J. Comput. Sci. Netw. Secur*, 17(4):160–165.

[Altun and Barshan, 2010] Altun, K. and Barshan, B. (2010). Human activity recognition using inertial/magnetic sensor units. In *Human Behavior Understanding: First International Workshop, HBU 2010, Istanbul, Turkey, August 22, 2010. Proceedings 1*, pages 38–51. Springer.

[Anguita et al., 2013] Anguita, D., Ghio, A., Oneto, L., Parra, X., and Reyes-Ortiz, J. L. (2013). A public domain dataset for human activity recognition using smartphones. In *Esann*, volume 3, page 3.

[Bachman et al., 2019] Bachman, P., Hjelm, R. D., and Buchwalter, W. (2019). Learning representations by maximizing mutual information across views. In *Advances in Neural Information Processing Systems*, pages 15535–15545.

[Bader et al., 2015] Bader, S., Krüger, F., and Kirste, T. (2015). Computational causal behaviour models for assisted manufacturing. In *Proceedings of the 2nd international Workshop on Sensor-based Activity Recognition and Interaction*, pages 1–6.

[Bai et al., 2018] Bai, S., Kolter, J. Z., and Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

[Baldi, 2012] Baldi, P. (2012). Autoencoders, unsupervised learning, and deep architectures. In Guyon, I., Dror, G., Lemaire, V., Taylor, G., and Silver, D., editors, *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, volume 27 of *Proceedings of Machine Learning Research*, pages 37–49, Bellevue, Washington, USA. PMLR.

[Bao and Intille, 2004] Bao, L. and Intille, S. S. (2004). Activity recognition from user-annotated acceleration data. In *International conference on pervasive computing*, pages 1–17. Springer.

[Bao et al., 2021] Bao, W., Yu, Q., and Kong, Y. (2021). Evidential deep learning for open set action recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13349–13358.

[Bardes et al., 2022] Bardes, A., Ponce, J., and Lecun, Y. (2022). Vicreg: Variance-invariance-covariance regularization for self-supervised learning. In *ICLR 2022-10th International Conference on Learning Representations*.

[Bayat et al., 2014] Bayat, A., Pomplun, M., and Tran, D. A. (2014). A study on human activity recognition using accelerometer data from smartphones. *Procedia Computer Science*, 34:450–457.

[Bennasar et al., 2022] Bennasar, M., Price, B. A., Gooch, D., Bandara, A. K., and Nuseibeh, B. (2022). Significant features for human activity recognition using tri-axial accelerometers. *Sensors*, 22(19):7482.

[Bento et al., 2022] Bento, N., Rebelo, J., Barandas, M., Carreiro, A. V., Campagner, A., Cabitza, F., and Gamboa, H. (2022). Comparing handcrafted features and deep neural representations for domain generalization in human activity recognition. *Sensors*, 22(19):7324.

[Bento et al., 2023] Bento, N., Rebelo, J., Carreiro, A. V., Ravache, F., and Barandas, M. (2023). Exploring regularization methods for domain generalization in accelerometer-based human activity recognition. *Sensors*, 23(14):6511.

[Bhattacharya et al., 2014] Bhattacharya, S., Nurmi, P., Hammerla, N., and Plötz, T. (2014). Using unlabeled data in a sparse-coding framework for human activity recognition. *Pervasive and Mobile Computing*, 15:242–262.

[Bommasani et al., 2021] Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. (2021). On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

[Bond-Taylor et al., 2021] Bond-Taylor, S., Leach, A., Long, Y., and Willcocks, C. G. (2021). Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models. *IEEE transactions on pattern analysis and machine intelligence*.

[Bulling et al., 2014] Bulling, A., Blanke, U., and Schiele, B. (2014). A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys (CSUR)*, 46(3):33.

[Buzzega et al., 2020] Buzzega, P., Boschini, M., Porrello, A., Abati, D., and Calderara, S. (2020). Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930.

[Caccia and Pineau, 2022] Caccia, L. and Pineau, J. (2022). Special: Self-supervised pretraining for continual learning. In *Continual Semi-Supervised Learning: First International Workshop, CSSL 2021, Virtual Event, August 19–20, 2021, Revised Selected Papers*, pages 91–103. Springer.

[Cai et al., 2020] Cai, Q., Wang, Y., Pan, Y., Yao, T., and Mei, T. (2020). Joint contrastive learning with infinite possibilities. *Advances in Neural Information Processing Systems*, 33:12638–12648.

[Caron et al., 2020] Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. (2020). Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems*, 33:9912–9924.

[Caron et al., 2021] Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., and Joulin, A. (2021). Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660.

[Caruana, 1997] Caruana, R. (1997). Multitask learning. *Machine learning*, 28(1):41–75.

[Casale et al., 2011] Casale, P., Pujol, O., and Radeva, P. (2011). Human activity recognition from accelerometer data using a wearable device. In *Pattern Recognition and*

Image Analysis: 5th Iberian Conference, IbPRIA 2011, Las Palmas de Gran Canaria, Spain, June 8-10, 2011. Proceedings 5, pages 289–296. Springer.

[Cha et al., 2021] Cha, H., Lee, J., and Shin, J. (2021). Co2l: Contrastive continual learning. In *Proceedings of the IEEE/CVF International conference on computer vision*, pages 9516–9525.

[Chang et al., 2020] Chang, Y., Mathur, A., Isopoussu, A., Song, J., and Kawsar, F. (2020). A systematic study of unsupervised domain adaptation for robust human-activity recognition. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(1):1–30.

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

[Chaudhry et al., 2019] Chaudhry, A., Ranzato, M., Rohrbach, M., and Elhoseiny, M. (2019). Efficient lifelong learning with A-GEM. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

[Chen et al., 2019] Chen, K., Yao, L., Zhang, D., Chang, X., Long, G., and Wang, S. (2019). Distributionally robust semi-supervised learning for people-centric sensing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3321–3328.

[Chen et al., 2021] Chen, K., Zhang, D., Yao, L., Guo, B., Yu, Z., and Liu, Y. (2021). Deep learning for sensor-based human activity recognition: Overview, challenges, and opportunities. *ACM Computing Surveys (CSUR)*, 54(4):1–40.

[Chen et al., 2024] Chen, S., Li, H., Wang, Q., Zhao, Z., Sun, M., Zhu, X., and Liu, J. (2024). Vast: A vision-audio-subtitle-text omni-modality foundation model and dataset. *Advances in Neural Information Processing Systems*, 36.

[Chen et al., 2020a] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020a). A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

[Chen et al., 2020b] Chen, X., Fan, H., Girshick, R., and He, K. (2020b). Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*.

[Chen and He, 2021] Chen, X. and He, K. (2021). Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758.

[Chen and Shen, 2017] Chen, Y. and Shen, C. (2017). Performance analysis of smartphone-sensor behavior for human activity recognition. *Ieee Access*, 5:3095–3110.

[Chen et al., 2016] Chen, Y., Zhong, K., Zhang, J., Sun, Q., and Zhao, X. (2016). Lstm networks for mobile human activity recognition. In *2016 International conference on artificial intelligence: technologies and applications*, pages 50–53. Atlantis Press.

[Cho et al., 2014] Cho, K., van Merriënboer, B., Gulçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

[Craighero et al., 2023] Craighero, M., Quarantiello, D., Rossi, B., Carrera, D., Fragneto, P., and Boracchi, G. (2023). On-device personalization for human activity recognition on stm32. *IEEE Embedded Systems Letters*.

[Dauphin et al., 2017] Dauphin, Y. N., Fan, A., Auli, M., and Grangier, D. (2017). Language modeling with gated convolutional networks. In *International conference on machine learning*, pages 933–941. PMLR.

[de la Concepción et al., 2017] de la Concepción, M. Á. Á., Morillo, L. M. S., García, J. A. Á., and González-Abril, L. (2017). Mobile activity recognition and fall detection system for elderly people using ameva algorithm. *Pervasive and Mobile Computing*, 34:3–13.

[De Lange et al., 2021] De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., and Tuytelaars, T. (2021). A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385.

[Deldari et al., 2022] Deldari, S., Xue, H., Saeed, A., Smith, D. V., and Salim, F. D. (2022). Cocoa: Cross modality contrastive learning for sensor data. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 6(3):1–28.

[Devlin et al., 2019] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for*

Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186.

[Dey et al., 2014] Dey, S., Roy, N., Xu, W., Choudhury, R. R., and Nelakuditi, S. (2014). Accelprint: Imperfections of accelerometers make smartphones trackable. In *NDSS*. Citeseer.

[Díaz-Rodríguez et al., 2018] Díaz-Rodríguez, N., Lomonaco, V., Filliat, D., and Maltoni, D. (2018). Don't forget, there is more than forgetting: new metrics for continual learning. In *Workshop on Continual Learning, NeurIPS 2018 (Neural Information Processing Systems*.

[Diethe et al., 2019] Diethe, T., Borchert, T., Thereska, E., Balle, B., and Lawrence, N. (2019). Continual learning in practice. *arXiv preprint arXiv:1903.05202*.

[Doersch et al., 2015] Doersch, C., Gupta, A., and Efros, A. A. (2015). Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430.

[Donahue et al., 2017] Donahue, J., Krähenbühl, P., and Darrell, T. (2017). Adversarial feature learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

[Ferrari et al., 2019] Ferrari, A., Micucci, D., Mobilio, M., and Napoletano, P. (2019). Human activities recognition using accelerometer and gyroscope. In *European Conference on Ambient Intelligence*, pages 357–362. Springer.

[Figo et al., 2010] Figo, D., Diniz, P. C., Ferreira, D. R., and Cardoso, J. M. (2010). Preprocessing techniques for context recognition from accelerometer data. *Personal and Ubiquitous Computing*, 14:645–662.

[Fini et al., 2022] Fini, E., da Costa, V. G. T., Alameda-Pineda, X., Ricci, E., Alahari, K., and Mairal, J. (2022). Self-supervised models are continual learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9621–9630.

[Floridi and Chiriatti, 2020] Floridi, L. and Chiriatti, M. (2020). Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30:681–694.

[Frosio et al., 2008] Frosio, I., Pedersini, F., and Borghese, N. A. (2008). Autocalibration of mems accelerometers. *IEEE Transactions on Instrumentation and Measurement*, 58(6):2034–2041.

[Gallardo et al., 2021] Gallardo, G. J., Hayes, T. L., and Kanan, C. (2021). Self-supervised training enhances online continual learning. In *32nd British Machine Vision Conference 2021, BMVC 2021, Online, November 22-25, 2021*, page 168. BMVA Press.

[Ganin and Lempitsky, 2015] Ganin, Y. and Lempitsky, V. (2015). Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR.

[Gardner and Dorling, 1998] Gardner, M. W. and Dorling, S. (1998). Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14-15):2627–2636.

[Ghayvat et al., 2015] Ghayvat, H., Liu, J., Mukhopadhyay, S. C., and Gui, X. (2015). Wellness sensor networks: A proposal and implementation for smart home for assisted living. *IEEE Sensors Journal*, 15(12):7341–7348.

[Girdhar et al., 2023] Girdhar, R., El-Nouby, A., Liu, Z., Singh, M., Alwala, K. V., Joulin, A., and Misra, I. (2023). Imagebind: One embedding space to bind them all. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15180–15190.

[Girshick, 2015] Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

[Goodfellow et al., 2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27.

[Goodfellow et al., 2013] Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A., and Bengio, Y. (2013). An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*.

[Grammenos et al., 2018] Grammenos, A., Mascolo, C., and Crowcroft, J. (2018). You are sensing, but are you biased?: A user unaided sensor calibration approach for mobile sensing. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 2(1):11:1–11:26.

[Graves et al., 2008] Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., and Schmidhuber, J. (2008). A novel connectionist system for unconstrained handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):855–868.

[Gretton et al., 2006] Gretton, A., Borgwardt, K., Rasch, M., Schölkopf, B., and Smola, A. (2006). A kernel method for the two-sample-problem. *Advances in neural information processing systems*, 19.

[Grill et al., 2020] Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al. (2020). Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284.

[Guan et al., 2007] Guan, D., Yuan, W., Lee, Y.-K., Gavrilov, A., and Lee, S. (2007). Activity recognition based on semi-supervised learning. In *13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2007)*, pages 469–475. IEEE.

[Habib and Qureshi, 2022] Habib, G. and Qureshi, S. (2022). Optimization and acceleration of convolutional neural networks: A survey. *Journal of King Saud University-Computer and Information Sciences*, 34(7):4244–4268.

[Hadsell et al., 2020] Hadsell, R., Rao, D., Rusu, A. A., and Pascanu, R. (2020). Embracing change: Continual learning in deep neural networks. *Trends in cognitive sciences*, 24(12):1028–1040.

[Hall et al., 2009] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

[Hammerla et al., 2013] Hammerla, N. Y., Kirkham, R., Andras, P., and Ploetz, T. (2013). On preserving statistical characteristics of accelerometry data using their empirical cumulative distribution. In *Proceedings of the 2013 international symposium on wearable computers*, pages 65–68.

[Han et al., 2022] Han, J., Xia, T., Spathis, D., Bondareva, E., Brown, C., Chauhan, J., Dang, T., Grammenos, A., Hasthanasombat, A., Floto, A., et al. (2022). Sounds of covid-19: exploring realistic performance of audio-based digital testing. *NPJ digital medicine*, 5(1):16.

[Haresamudram et al., 2019] Haresamudram, H., Anderson, D. V., and Plötz, T. (2019). On the role of features in human activity recognition. In *Proceedings of the 2019 ACM International Symposium on Wearable Computers*, pages 78–88.

[Haresamudram et al., 2020] Haresamudram, H., Beedu, A., Agrawal, V., Grady, P. L., Essa, I., Hoffman, J., and Plötz, T. (2020). Masked reconstruction based self-supervision for human activity recognition. In *Proceedings of the 2020 ACM International Symposium on Wearable Computers*, pages 45–49.

[Haresamudram et al., 2021] Haresamudram, H., Essa, I., and Plötz, T. (2021). Contrastive predictive coding for human activity recognition. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 5(2):1–26.

[Haresamudram et al., 2022] Haresamudram, H., Essa, I., and Plötz, T. (2022). Assessing the state of self-supervised human activity recognition using wearables. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 6(3):1–47.

[Hasthanasombat et al., 2022] Hasthanasombat, A., Ghosh, A., Spathis, D., and Mascolo, C. (2022). Investigating domain-agnostic performance in activity recognition using accelerometer data. In *Adjunct Proceedings of the 2022 ACM International Joint Conference on Pervasive and Ubiquitous Computing and the 2022 ACM International Symposium on Wearable Computers*, pages 329–334.

[Hayes et al., 2020] Hayes, T. L., Kafle, K., Shrestha, R., Acharya, M., and Kanan, C. (2020). Remind your neural network to prevent catastrophic forgetting. In *European Conference on Computer Vision*, pages 466–483. Springer.

[He et al., 2020] He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738.

[Hinton et al., 2015] Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Hou et al., 2019] Hou, S., Pan, X., Loy, C. C., Wang, Z., and Lin, D. (2019). Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 831–839.

[Hsu et al., 2018] Hsu, Y.-C., Liu, Y.-C., Ramasamy, A., and Kira, Z. (2018). Re-evaluating continual learning scenarios: A categorization and case for strong baselines. *arXiv preprint arXiv:1810.12488*.

[Hu et al., 2023] Hu, R., Chen, L., Miao, S., and Tang, X. (2023). Swl-adapt: An unsupervised domain adaptation model with sample weight learning for cross-user wearable human activity recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 6012–6020.

[Hucklesby and Holt, 2023] Hucklesby, A. and Holt, R. (2023). *Tracking People: Wearable Technologies in Social and Public Policy*. Taylor & Francis.

[Huynh and Schiele, 2005] Huynh, T. and Schiele, B. (2005). Analyzing features for activity recognition. In *Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies*, pages 159–163.

[Ignatov, 2018] Ignatov, A. (2018). Real-time human activity recognition from accelerometer data using convolutional neural networks. *Applied Soft Computing*, 62:915–922.

[Iscen et al., 2020] Iscen, A., Zhang, J., Lazebnik, S., and Schmid, C. (2020). Memory-efficient incremental learning through feature adaptation. In *European Conference on Computer Vision*, pages 699–715. Springer.

[Isele and Cosgun, 2018] Isele, D. and Cosgun, A. (2018). Selective experience replay for lifelong learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

[Jacobs, 2005] Jacobs, D. (2005). Correlation and convolution. *Class Notes for CMSC*, 426:401–409.

[Jain et al., 2022] Jain, Y., Tang, C. I., Min, C., Kawsar, F., and Mathur, A. (2022). Collossl: Collaborative self-supervised learning for human activity recognition. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 6(1):1–28.

[Jalal et al., 2014] Jalal, A., Kamal, S., and Kim, D. (2014). A depth video sensor-based life-logging human activity recognition system for elderly care in smart indoor environments. *Sensors*, 14(7):11735–11759.

[Jiang et al., 2018] Jiang, W., Miao, C., Ma, F., Yao, S., Wang, Y., Yuan, Y., Xue, H., Song, C., Ma, X., Koutsonikolas, D., et al. (2018). Towards environment independent device free human activity recognition. In *Proceedings of the 24th annual international conference on mobile computing and networking*, pages 289–304.

[Jing and Tian, 2020] Jing, L. and Tian, Y. (2020). Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Kalouris et al., 2019] Kalouris, G., Zacharaki, E. I., and Megalooikonomou, V. (2019). Improving cnn-based activity recognition by data augmentation and transfer learning. In *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, volume 1, pages 1387–1394. IEEE.

[Kang et al., 2010] Kang, S., Lee, Y., Min, C., Ju, Y., Park, T., Lee, J., Rhee, Y., and Song, J. (2010). Orchestrator: An active resource orchestration framework for mobile context monitoring in sensor-rich mobile environments. In *2010 ieee international conference on pervasive computing and communications (percom)*, pages 135–144. IEEE.

[Keally et al., 2011] Keally, M., Zhou, G., Xing, G., Wu, J., and Pyles, A. (2011). Pbn: towards practical activity recognition using smartphone-based body sensor networks. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, pages 246–259.

[Kingma and Ba, 2015] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.*

[Kingma et al., 2014] Kingma, D. P., Mohamed, S., Rezende, D. J., and Welling, M. (2014). Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pages 3581–3589.

[Kirkpatrick et al., 2017] Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

[Kunze and Lukowicz, 2008] Kunze, K. and Lukowicz, P. (2008). Dealing with sensor displacement in motion-based onbody activity recognition systems. In *Proceedings of the 10th international conference on Ubiquitous computing*, pages 20–29.

[Kwapisz et al., 2011] Kwapisz, J. R., Weiss, G. M., and Moore, S. A. (2011). Activity recognition using cell phone accelerometers. *ACM SigKDD Explorations Newsletter*, 12(2):74–82.

[Kwon et al., 2018] Kwon, H., Abowd, G. D., and Plötz, T. (2018). Adding structural characteristics to distribution-based accelerometer representations for activity recognition using wearables. In *Proceedings of the 2018 ACM international symposium on wearable computers*, pages 72–75.

[Kwon et al., 2020] Kwon, H., Tong, C., Haresamudram, H., Gao, Y., Abowd, G. D., Lane, N. D., and Ploetz, T. (2020). Imutube: Automatic extraction of virtual on-body accelerometry from video for human activity recognition. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(3):1–29.

[Ladha et al., 2013] Ladha, C., Hammerla, N. Y., Olivier, P., and Plötz, T. (2013). Climbax: skill assessment for climbing enthusiasts. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 235–244. ACM.

[Lara and Labrador, 2013] Lara, O. D. and Labrador, M. A. (2013). A survey on human activity recognition using wearable sensors. *IEEE communications surveys & tutorials*, 15(3):1192–1209.

[Laskaridis et al., 2021] Laskaridis, S., Kouris, A., and Lane, N. D. (2021). Adaptive inference through early-exit networks: Design, challenges and directions. In *Proceedings of the 5th International Workshop on Embedded and Mobile Deep Learning*, pages 1–6.

[LeCun et al., 1989] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

[Ledig et al., 2017] Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al. (2017). Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690.

[Lee et al., 2017] Lee, S.-M., Yoon, S. M., and Cho, H. (2017). Human activity recognition from accelerometer data using convolutional neural network. In *2017 ieee international conference on big data and smart computing (bigcomp)*, pages 131–134. IEEE.

[Lemaréchal, 2012] Lemaréchal, C. (2012). Cauchy and the gradient method. *Doc Math Extra*, 251(254):10.

[Leng et al., 2023] Leng, Z., Kwon, H., and Plötz, T. (2023). Generating virtual on-body accelerometer data from virtual textual descriptions for human activity recognition. In *Proceedings of the 2023 ACM International Symposium on Wearable Computers*, pages 39–43.

[Li et al., 2019] Li, H., Li, J., Guan, X., Liang, B., Lai, Y., and Luo, X. (2019). Research on overfitting of deep learning. In *2019 15th international conference on computational intelligence and security (CIS)*, pages 78–81. IEEE.

[Li et al., 2020] Li, X., Luo, J., and Younes, R. (2020). Activitygan: Generative adversarial networks for data augmentation in sensor-based human activity recognition. In

Adjunct Proceedings of the 2020 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2020 ACM International Symposium on Wearable Computers, pages 249–254.

[Li et al., 2016] Li, Y., Paluri, M., Rehg, J. M., and Dollár, P. (2016). Unsupervised learning of edges. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1619–1627.

[Li and Hoiem, 2017] Li, Z. and Hoiem, D. (2017). Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947.

[Liang et al., 2014] Liang, Y., Zhou, X., Yu, Z., and Guo, B. (2014). Energy-efficient motion related activity recognition on mobile devices for pervasive healthcare. *Mobile Networks and Applications*, 19(3):303–317.

[Lin et al., 2021] Lin, J., Fu, R., Zhong, X., Yu, P., Tan, G., Li, W., Zhang, H., Li, Y., Zhou, L., and Ning, C. (2021). Wearable sensors and devices for real-time cardiovascular disease monitoring. *Cell reports physical science*, 2(8).

[Liono et al., 2016] Liono, J., Qin, A. K., and Salim, F. D. (2016). Optimal time window for temporal segmentation of sensor streams in multi-activity recognition. In *Proceedings of the 13th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pages 10–19.

[Lopez-Paz and Ranzato, 2017] Lopez-Paz, D. and Ranzato, M. (2017). Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30:6467–6476.

[Lotta et al., 2018] Lotta, L. A., Wittemans, L. B., Zuber, V., Stewart, I. D., Sharp, S. J., Luan, J., Day, F. R., Li, C., Bowker, N., Cai, L., et al. (2018). Association of genetic variants related to gluteofemoral vs abdominal fat distribution with type 2 diabetes, coronary disease, and cardiovascular risk factors. *Jama*, 320(24):2553–2563.

[Ma et al., 2022] Ma, D., Tang, C. I., and Mascolo, C. (2022). Improving feature generalizability with multitask learning in class incremental learning. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4173–4177. IEEE.

[Maaten and Hinton, 2008] Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

[Madaan et al., 2021] Madaan, D., Yoon, J., Li, Y., Liu, Y., and Hwang, S. J. (2021). Rethinking the representational continuity: Towards unsupervised continual learning. *arXiv preprint arXiv:2110.06976*.

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

[Martín et al., 2013] Martín, H., Bernardos, A. M., Iglesias, J., and Casar, J. R. (2013). Activity logging using lightweight classification techniques in mobile devices. *Personal and ubiquitous computing*, 17:675–695.

[Masum et al., 2019] Masum, A. K. M., Bahadur, E. H., Shan-A-Alahi, A., Chowdhury, M. A. U. Z., Uddin, M. R., and Al Noman, A. (2019). Human activity recognition using accelerometer, gyroscope and magnetometer sensors: Deep neural network approaches. In *2019 10Th international conference on computing, communication and networking technologies (ICCCNT)*, pages 1–6. IEEE.

[Mathur et al., 2018] Mathur, A., Zhang, T., Bhattacharya, S., Velickovic, P., Joffe, L., Lane, N. D., Kawsar, F., and Lió, P. (2018). Using deep data augmentation training to address software and hardware heterogeneities in wearable and smartphone sensing devices. In *2018 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 200–211. IEEE.

[Matsui et al., 2017] Matsui, S., Inoue, N., Akagi, Y., Nagino, G., and Shinoda, K. (2017). User adaptation of convolutional neural network for human activity recognition. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 753–757. IEEE.

[McCulloch and Pitts, 1943] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133.

[Medsker and Jain, 1999] Medsker, L. and Jain, L. C. (1999). *Recurrent neural networks: design and applications*. CRC press.

[Mehrabi et al., 2021] Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., and Galstyan, A. (2021). A survey on bias and fairness in machine learning. *ACM computing surveys (CSUR)*, 54(6):1–35.

[Menschner et al., 2011] Menschner, P., Prinz, A., Koene, P., Köbler, F., Altmann, M., Krcmar, H., and Leimeister, J. M. (2011). Reaching into patients' homes–participatory designed aal services. *Electronic Markets*, 21(1):63–76.

[Merrill and Althoff, 2023] Merrill, M. A. and Althoff, T. (2023). Self-supervised pretraining and transfer learning enable flu and covid-19 predictions in small mobile sensing datasets. In *Conference on Health, Inference, and Learning*, pages 191–206. PMLR.

[Merrill et al., 2023] Merrill, M. A., Safranchik, E., Kolbeinsson, A., Gade, P., Ramirez, E., Schmidt, L., Foshchini, L., and Althoff, T. (2023). Homekit2020: A benchmark for time series classification on a large mobile sensing dataset with laboratory tested ground truth of influenza infections. In *Conference on Health, Inference, and Learning*, pages 207–228. PMLR.

[Mezlini et al., 2022] Mezlini, A., Shapiro, A., Daza, E. J., Caddigan, E., Ramirez, E., Althoff, T., and Foschini, L. (2022). Estimating the Burden of Influenza-like Illness on Daily Activity at the Population Scale Using Commercial Wearable Sensors. *JAMA Network Open*, 5(5):e2211958–e2211958.

[Micucci et al., 2017] Micucci, D., Mobilio, M., and Napoletano, P. (2017). Unimib shar: A dataset for human activity recognition using acceleration data from smartphones. *Applied Sciences*, 7(10):1101.

[Min et al., 2019] Min, C., Montanari, A., Mathur, A., and Kawsar, F. (2019). A closer look at quality-aware runtime assessment of sensing models in multi-device environments. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*, pages 271–284.

[Misra et al., 2016] Misra, I., Zitnick, C. L., and Hebert, M. (2016). Shuffle and learn: unsupervised learning using temporal order verification. In *European Conference on Computer Vision*, pages 527–544. Springer.

[Mittal et al., 2021] Mittal, S., Galesso, S., and Brox, T. (2021). Essentials for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3513–3522.

[Moeslund et al., 2006] Moeslund, T. B., Hilton, A., and Krüger, V. (2006). A survey of advances in vision-based human motion capture and analysis. *Computer vision and image understanding*, 104(2-3):90–126.

[Moon et al., 2022] Moon, S., Madotto, A., Lin, Z., Dirafzoon, A., Saraf, A., Bearman, A., and Damavandi, B. (2022). Imu2clip: Multimodal contrastive learning for imu motion sensors from egocentric videos and text. *arXiv preprint arXiv:2210.14395*.

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

[Murad et al., 2020] Murad, A., Kraemer, F. A., Bach, K., and Taylor, G. (2020). Information-driven adaptive sensing based on deep reinforcement learning. In *Proceedings of the 10th International Conference on the Internet of Things*, pages 1–8.

[Noroozi and Favaro, 2016] Noroozi, M. and Favaro, P. (2016). Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer.

[Nweke et al., 2018] Nweke, H. F., Teh, Y. W., Al-Garadi, M. A., and Alo, U. R. (2018). Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges. *Expert Systems with Applications*, 105:233–261.

[Oord et al., 2018] Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

[Oquab et al., 2014] Oquab, M., Bottou, L., Laptev, I., and Sivic, J. (2014). Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1717–1724.

[Ordóñez and Roggen, 2016] Ordóñez, F. J. and Roggen, D. (2016). Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16(1):115.

[Ostapenko et al., 2019] Ostapenko, O., Puscas, M., Klein, T., Jahnichen, P., and Nabi, M. (2019). Learning to remember: A synaptic plasticity driven framework for continual learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11321–11329.

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

[Park et al., 2019] Park, D. S., Chan, W., Zhang, Y., Chiu, C., Zoph, B., Cubuk, E. D., and Le, Q. V. (2019). Specaugment: A simple data augmentation method for automatic speech recognition. In Kubin, G. and Kacic, Z., editors, *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019*, pages 2613–2617. ISCA.

[Paszke et al., 2019] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

[Pathak et al., 2016] Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., and Efros, A. A. (2016). Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544.

[Peng et al., 2018] Peng, L., Chen, L., Ye, Z., and Zhang, Y. (2018). Aroma: A deep multi-task learning based simple and complex human activity recognition method using wearable sensors. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(2):1–16.

[Pienaar and Malekian, 2019] Pienaar, S. W. and Malekian, R. (2019). Human activity recognition using lstm-rnn deep neural network architecture. In *2019 IEEE 2nd wireless africa conference (WAC)*, pages 1–5. IEEE.

[Pironkov et al., 2016] Pironkov, G., Dupont, S., and Dutoit, T. (2016). Speaker-aware long short-term memory multi-task learning for speech recognition. In *2016 24th European Signal Processing Conference (EUSIPCO)*, pages 1911–1915. IEEE.

[Plötz, 2021] Plötz, T. (2021). Applying machine learning for sensor data analysis in interactive systems: Common pitfalls of pragmatic use and ways to avoid them. *ACM Computing Surveys (CSUR)*, 54(6):1–25.

[Plötz et al., 2011] Plötz, T., Hammerla, N. Y., and Olivier, P. L. (2011). Feature learning for activity recognition in ubiquitous computing. In *Twenty-second international joint conference on artificial intelligence*.

[Poddar et al., 2017] Poddar, S., Kumar, V., and Kumar, A. (2017). A comprehensive overview of inertial sensor calibration techniques. *Journal of Dynamic Systems, Measurement, and Control*, 139(1).

[Qin et al., 2022] Qin, X., Wang, J., Chen, Y., Lu, W., and Jiang, X. (2022). Domain generalization for activity recognition via adaptive feature fusion. *ACM Transactions on Intelligent Systems and Technology*, 14(1):1–21.

[Radford et al., 2021] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

[Radford et al., 2016] Radford, A., Metz, L., and Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks. In Bengio, Y. and LeCun, Y., editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

[Ranasinghe et al., 2016] Ranasinghe, S., Al Machot, F., and Mayr, H. C. (2016). A review on applications of activity recognition systems with regard to performance and evaluation. *International Journal of Distributed Sensor Networks*, 12(8):1550147716665520.

[Rebuffi et al., 2017] Rebuffi, S.-A., Kolesnikov, A., Sperl, G., and Lampert, C. H. (2017). icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010.

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

[Roggen et al., 2010] Roggen, D., Calatroni, A., Rossi, M., Holleczek, T., Förster, K., Tröster, G., Lukowicz, P., Bannach, D., Pirkl, G., Ferscha, A., et al. (2010). Collecting complex activity datasets in highly rich networked sensor environments. In *2010 Seventh international conference on networked sensing systems (INSS)*, pages 233–240. IEEE.

[Rolnick et al., 2019] Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., and Wayne, G. (2019). Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32.

[Ronao and Cho, 2016] Ronao, C. A. and Cho, S.-B. (2016). Human activity recognition with smartphone sensors using deep learning neural networks. *Expert systems with applications*, 59:235–244.

[Rosenberg et al., 2005] Rosenberg, C., Hebert, M., and Schneiderman, H. (2005). Semi-supervised self-training of object detection models.

[Rosenblatt, 1958] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

[Rosenblatt et al., 1962] Rosenblatt, F. et al. (1962). *Principles of neurodynamics: Perceptrons and the theory of brain mechanisms*, volume 55. Spartan books Washington, DC.

[Rovini et al., 2017] Rovini, E., Maremmani, C., and Cavallo, F. (2017). How wearable sensors can support parkinson's disease diagnosis and treatment: a systematic review. *Frontiers in neuroscience*, 11:555.

[Ruder, 2017] Ruder, S. (2017). An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

[Rusu et al., 2016] Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. (2016). Progressive neural networks. *arXiv preprint arXiv:1606.04671*.

[Sabokrou et al., 2018] Sabokrou, M., Fayyaz, M., Fathy, M., Moayed, Z., and Klette, R. (2018). Deep-anomaly: Fully convolutional neural network for fast anomaly detection in crowded scenes. *Computer Vision and Image Understanding*, 172:88–97.

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

[Saeed et al., 2021] Saeed, A., Ungureanu, V., and Gfeller, B. (2021). Sense and learn: Self-supervision for omnipresent sensors. *Machine Learning with Applications*, 6:100152.

[Safaei et al., 2017] Safaei, B., Monazzah, A. M. H., Bafroei, M. B., and Ejlali, A. (2017). Reliability side-effects in internet of things application layer protocols. In *2017 2nd International Conference on System Reliability and Safety (ICSRS)*, pages 207–212. IEEE.

[Sagha et al., 2011] Sagha, H., Digumarti, S. T., Millán, J. d. R., Chavarriaga, R., Calatroni, A., Roggen, D., and Tröster, G. (2011). Benchmarking classification techniques using the opportunity human activity dataset. In *2011 IEEE International Conference on Systems, Man, and Cybernetics*, pages 36–40. IEEE.

[Sak et al., 2014] Sak, H., Senior, A. W., and Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling.

[Sani et al., 2018] Sani, S., Wiratunga, N., Massie, S., and Cooper, K. (2018). Personalised human activity recognition using matching networks. In *Case-Based Reasoning Research and Development: 26th International Conference, ICCBR 2018, Stockholm, Sweden, July 9-12, 2018, Proceedings 26*, pages 339–353. Springer.

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

[Serra et al., 2018] Serra, J., Suris, D., Miron, M., and Karatzoglou, A. (2018). Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, pages 4548–4557. PMLR.

[Shi et al., 2021] Shi, H., Zhang, Y., Shen, Z., Tang, S., Li, Y., Guo, Y., and Zhuang, Y. (2021). Federated self-supervised contrastive learning via ensemble similarity distillation. *arXiv preprint arXiv:2109.14611*.

[Shin et al., 2017] Shin, H., Lee, J. K., Kim, J., and Kim, J. (2017). Continual learning with deep generative replay. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 2994–3003.

[Shoaib et al., 2016] Shoaib, M., Bosch, S., Incel, O. D., Scholten, H., and Havinga, P. J. (2016). Complex human activity recognition using smartphone and wrist-worn motion sensors. *Sensors*, 16(4):426.

[Shu et al., 2018] Shu, Y., Shi, Y., Wang, Y., Zou, Y., Yuan, Q., and Tian, Y. (2018). Odn: Opening the deep network for open-set action recognition. In *2018 IEEE international conference on multimedia and expo (ICME)*, pages 1–6. IEEE.

[Simonyan et al., 2014] Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Deep inside convolutional networks: Visualising image classification models and saliency maps. In Bengio, Y. and LeCun, Y., editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings*.

[Spathis et al., 2021] Spathis, D., Perez-Pozuelo, I., Brage, S., Wareham, N. J., and Mascolo, C. (2021). Self-supervised transfer learning of physiological representations from free-living wearable data. In *Proceedings of the Conference on Health, Inference, and Learning*, pages 69–78.

[Stahlberg, 2020] Stahlberg, F. (2020). Neural machine translation: A review. *Journal of Artificial Intelligence Research*, 69:343–418.

[Stikic et al., 2008] Stikic, M., Van Laerhoven, K., and Schiele, B. (2008). Exploring semi-supervised and active learning for activity recognition. In *2008 12th IEEE International Symposium on Wearable Computers*, pages 81–88. IEEE.

[Stisen et al., 2015] Stisen, A., Blunck, H., Bhattacharya, S., Prentow, T. S., Kjærgaard, M. B., Dey, A., Sonne, T., and Jensen, M. M. (2015). Smart devices are different: Assessing and mitigating mobile sensing heterogeneities for activity recognition. In *Proceedings of the 13th ACM conference on embedded networked sensor systems*, pages 127–140.

[Sudlow et al., 2015] Sudlow, C., Gallacher, J., Allen, N., Beral, V., Burton, P., Danesh, J., Downey, P., Elliott, P., Green, J., Landray, M., et al. (2015). Uk biobank: an open access resource for identifying the causes of a wide range of complex diseases of middle and old age. *PLoS medicine*, 12(3):e1001779.

[Sun et al., 2019] Sun, Q., Liu, Y., Chua, T.-S., and Schiele, B. (2019). Meta-transfer learning for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 403–412.

[Sutton and Barto, 2018] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction.* MIT press.

[Sztyler and Stuckenschmidt, 2016] Sztyler, T. and Stuckenschmidt, H. (2016). On-body localization of wearable devices: An investigation of position-aware activity recognition. In *2016 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–9. IEEE.

[Tan et al., 2018] Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., and Liu, C. (2018). A survey on deep transfer learning. In *International conference on artificial neural networks*, pages 270–279. Springer.

[Tang et al., 2021] Tang, C. I., Perez-Pozuelo, I., Spathis, D., Brage, S., Wareham, N., and Mascolo, C. (2021). Selfhar: Improving human activity recognition through self-training with unlabeled data. *Proceedings of the ACM on interactive, mobile, wearable and ubiquitous technologies*, 5(1):1–30.

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

[Tang et al., 2024a] Tang, C. I., Qendro, L., Spathis, D., Kawsar, F., Mascolo, C., and Mathur, A. (2024a). Kaizen: Practical self-supervised continual learning with continual fine-tuning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2841–2850.

[Tang et al., 2024b] Tang, C. I., Qendro, L., Spathis, D., Kawsar, F., Mathur, A., and Mascolo, C. (2024b). Balancing continual learning and fine-tuning for human activity recognition. *arXiv preprint arXiv:2401.02255*.

[Thakur et al., 2022] Thakur, D., Biswas, S., Ho, E. S., and Chattopadhyay, S. (2022). Convae-lstm: Convolutional autoencoder long short-term memory network for smartphone-based human activity recognition. *IEEE Access*, 10:4137–4156.

[Tommasi et al., 2017] Tommasi, T., Patricia, N., Caputo, B., and Tuytelaars, T. (2017). A deeper look at dataset bias. In *Domain adaptation in computer vision applications*, pages 37–55. Springer.

[Tong et al., 2020] Tong, C., Tailor, S. A., and Lane, N. D. (2020). Are accelerometers for activity recognition a dead-end? In *Proceedings of the 21st International Workshop on Mobile Computing Systems and Applications*, pages 39–44.

[Torralba and Efros, 2011] Torralba, A. and Efros, A. A. (2011). Unbiased look at dataset bias. In *CVPR 2011*, pages 1521–1528. IEEE.

[Touvron et al., 2023] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. (2023). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

[Um et al., 2017] Um, T. T., Pfister, F. M., Pichler, D., Endo, S., Lang, M., Hirche, S., Fietzek, U., and Kulić, D. (2017). Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, pages 216–220.

[Vaizman et al., 2017] Vaizman, Y., Ellis, K., and Lanckriet, G. (2017). Recognizing detailed human context in the wild from smartphones and smartwatches. *IEEE Pervasive Computing*, 16(4):62–74.

[Vaizman et al., 2018] Vaizman, Y., Weibel, N., and Lanckriet, G. (2018). Context recognition in-the-wild: Unified model for multi-modal sensors and multi-label classification. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(4):1–22.

[Van de Ven and Tolias, 2019] Van de Ven, G. M. and Tolias, A. S. (2019). Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*.

[van den Oord et al., 2016] van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. W., and Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. In *The 9th ISCA Speech Synthesis Workshop, Sunnyvale, CA, USA, 13-15 September 2016*, page 125. ISCA.

[Van Engelen and Hoos, 2020] Van Engelen, J. E. and Hoos, H. H. (2020). A survey on semi-supervised learning. *Machine Learning*, 109(2):373–440.

[van Hees et al., 2013] van Hees, V. T., Golubic, R., Ekelund, U., and Brage, S. (2013). Impact of study design on development and evaluation of an activity-type classifier. *Journal of Applied Physiology*, 114(8):1042–1051.

[Varamin et al., 2018] Varamin, A. A., Abbasnejad, E., Shi, Q., Ranasinghe, D. C., and Rezatofighi, H. (2018). Deep auto-set: A deep auto-encoder-set network for activity recognition using wearables. In *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pages 246–253.

[Vincent et al., 2008] Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103.

[Waibel, 1989] Waibel, A. (1989). Modular construction of time-delay neural networks for speech recognition. *Neural computation*, 1(1):39–46.

[Wang et al., 2022] Wang, J., Chen, D., Wu, Z., Luo, C., Zhou, L., Zhao, Y., Xie, Y., Liu, C., Jiang, Y.-G., and Yuan, L. (2022). Omnivl: One foundation model for image-language and video-language tasks. *Advances in neural information processing systems*, 35:5696–5710.

[Wang et al., 2018] Wang, J., Chen, Y., Gu, Y., Xiao, Y., and Pan, H. (2018). Sensorygans: an effective generative adversarial framework for sensor-based human activity recognition. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

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

[Wang et al., 2009] Wang, Y., Lin, J., Annavaram, M., Jacobson, Q. A., Hong, J., Krishnamachari, B., and Sadeh, N. (2009). A framework of energy efficient mobile sensing for automatic user state recognition. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, pages 179–192.

[Webber and Rojas, 2021] Webber, M. and Rojas, R. F. (2021). Human activity recognition with accelerometer and gyroscope: A data fusion approach. *IEEE Sensors Journal*, 21(15):16979–16989.

[Wei et al., 2019] Wei, C., Xie, L., Ren, X., Xia, Y., Su, C., Liu, J., Tian, Q., and Yuille, A. L. (2019). Iterative reorganization with weak spatial constraints: Solving arbitrary jigsaw puzzles for unsupervised representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1910–1919.

[Weiss, 2019] Weiss, G. M. (2019). Wisdm smartphone and smartwatch activity and biometrics dataset. *UCI Machine Learning Repository: WISDM Smartphone and Smartwatch Activity and Biometrics Dataset Data Set*, 7:133190–133202.

[White et al., 2016] White, T., Westgate, K., Wareham, N. J., and Brage, S. (2016). Estimation of physical activity energy expenditure during free-living from wrist accelerometry in uk adults. *PLoS One*, 11(12):e0167472.

[Wilson et al., 2020] Wilson, G., Doppa, J. R., and Cook, D. J. (2020). Multi-source deep domain adaptation with weak supervision for time-series sensor data. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1768–1778.

[Wu et al., 2019] Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., and Fu, Y. (2019). Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 374–382.

[Xia et al., 2020] Xia, Q., Korpela, J., Namioka, Y., and Maekawa, T. (2020). Robust unsupervised factory activity recognition with body-worn accelerometer using temporal structure of multiple sensor data motifs. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(3):1–30.

[Yalniz et al., 2019] Yalniz, I. Z., Jégou, H., Chen, K., Paluri, M., and Mahajan, D. (2019). Billion-scale semi-supervised learning for image classification. *arXiv preprint arXiv:1905.00546*.

[Yan et al., 2012] Yan, Z., Subbaraju, V., Chakraborty, D., Misra, A., and Aberer, K. (2012). Energy-efficient continuous activity recognition on mobile phones: An activity-adaptive approach. In *Wearable Computers (ISWC), 2012 16th International Symposium on*, pages 17–24. Ieee.

[Yang et al., 2015] Yang, J., Nguyen, M. N., San, P. P., Li, X., and Krishnaswamy, S. (2015). Deep convolutional neural networks on multichannel time series for human activity recognition. In *Ijcai*, volume 15, pages 3995–4001. Buenos Aires, Argentina.

[Yang et al., 2007] Yang, J.-Y., Chen, Y.-P., Lee, G.-Y., Liou, S.-N., and Wang, J.-S. (2007). Activity recognition using one triaxial accelerometer: A neuro-fuzzy classifier with feature reduction. In *International conference on entertainment computing*, pages 395–400. Springer.

[Yang et al., 2019] Yang, Y., Hou, C., Lang, Y., Guan, D., Huang, D., and Xu, J. (2019). Open-set human activity recognition based on micro-doppler signatures. *Pattern Recognition*, 85:60–69.

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

[Yao et al., 2018a] Yao, S., Zhao, Y., Hu, S., and Abdelzaher, T. (2018a). Qualitydeepsense: Quality-aware deep learning framework for internet of things applications with sensor-temporal attention. In *Proceedings of the 2nd International Workshop on Embedded and Mobile Deep Learning*, pages 42–47.

[Yao et al., 2018b] Yao, S., Zhao, Y., Shao, H., Zhang, C., Zhang, A., Hu, S., Liu, D., Liu, S., Su, L., and Abdelzaher, T. (2018b). Sensegan: Enabling deep learning for internet of things with a semi-supervised framework. *Proceedings of the ACM on interactive, mobile, wearable and ubiquitous technologies*, 2(3):1–21.

[Yarowsky, 1995] Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics*, pages 189–196.

[Yoon et al., 2019] Yoon, J., Jarrett, D., and Van der Schaar, M. (2019). Time-series generative adversarial networks. *Advances in neural information processing systems*, 32.

[Yosinski et al., 2014] Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328.

[Zappi et al., 2008] Zappi, P., Lombriser, C., Stiefmeier, T., Farella, E., Roggen, D., Benini, L., and Tröster, G. (2008). Activity recognition from on-body sensors: accuracy-power trade-off by dynamic sensor selection. In *Wireless sensor networks*, pages 17–33. Springer.

[Zaremba et al., 2014] Zaremba, W., Sutskever, I., and Vinyals, O. (2014). Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

[Zbontar et al., 2021] Zbontar, J., Jing, L., Misra, I., LeCun, Y., and Deny, S. (2021). Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, pages 12310–12320. PMLR.

[Zeng et al., 2014] Zeng, M., Nguyen, L. T., Yu, B., Mengshoel, O. J., Zhu, J., Wu, P., and Zhang, J. (2014). Convolutional neural networks for human activity recognition using mobile sensors. In *6th international conference on mobile computing, applications and services*, pages 197–205. IEEE.

[Zenke et al., 2017] Zenke, F., Poole, B., and Ganguli, S. (2017). Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995. PMLR.

[Zhai et al., 2020] Zhai, B., Perez-Pozuelo, I., Clifton, E. A., Palotti, J., and Guan, Y. (2020). Making sense of sleep: Multimodal sleep stage classification in a large, diverse population using movement and cardiac sensing. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(2):1–33.

[Zhang et al., 2016] Zhang, R., Isola, P., and Efros, A. A. (2016). Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer.

[Zhang et al., 1988] Zhang, W., Tanida, J., Itoh, K., and Ichioka, Y. (1988). Shift-invariant pattern recognition neural network and its optical architecture. In *Proceedings of annual conference of the Japan Society of Applied Physics*, volume 564. Montreal, CA.

[Zhang et al., 2022] Zhang, X., Zhao, Z., Tsiligkaridis, T., and Zitnik, M. (2022). Self-supervised contrastive pre-training for time series via time-frequency consistency. *Advances in Neural Information Processing Systems*, 35:3988–4003.

[Zhang and Yang, 2018] Zhang, Y. and Yang, Q. (2018). An overview of multi-task learning. *National Science Review*, 5(1):30–43.

[Zhang and Yang, 2021] Zhang, Y. and Yang, Q. (2021). A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1.

[Zhao et al., 2020] Zhao, B., Xiao, X., Gan, G., Zhang, B., and Xia, S.-T. (2020). Maintaining discrimination and fairness in class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13208–13217.

[Zhou et al., 2019] Zhou, P., Mai, L., Zhang, J., Xu, N., Wu, Z., and Davis, L. S. (2019). M2kd: Multi-model and multi-level knowledge distillation for incremental learning. *arXiv preprint arXiv:1904.01769*.

[Zhou et al., 2020] Zhou, Z., Zhang, Y., Yu, X., Yang, P., Li, X.-Y., Zhao, J., and Zhou, H. (2020). Xhar: Deep domain adaptation for human activity recognition with smart devices. In *2020 17th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 1–9. IEEE.

[Zhu and Goldberg, 2009] Zhu, X. and Goldberg, A. B. (2009). Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130.

[Zhu, 2005] Zhu, X. J. (2005). Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.

[Zou et al., 2018] Zou, Y., Yu, Z., Vijaya Kumar, B., and Wang, J. (2018). Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *Proceedings of the European conference on computer vision (ECCV)*, pages 289–305.

[Zuo et al., 2023] Zuo, S., Rey, V. F., Suh, S., Sigg, S., and Lukowicz, P. (2023). Unsupervised statistical feature-guided diffusion model for sensor-based human activity recognition. *arXiv preprint arXiv:2306.05285*.