ENHANCING LIGHTWEIGHT MODELS FOR EFFICIENT SENSOR-BASED HUMAN ACTIVITY RECOGNITION

by

Sümeyye Ağaç

B.S., Computer Engineering, Galatasaray University, 2016

M.S., Computer Engineering, Galatasaray University, 2019

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Computer Engineering

Boğaziçi University

2024

ACKNOWLEDGEMENTS

In this thesis, focused on creating resource-efficient models, I worked diligently to make the best use of my time and energy, just as I did with the tiny devices. Throughout this journey, I was fortunate to receive support from many people, to whom I am deeply grateful.

I want to extend my deepest gratitude to my supervisor, Özlem Durmaz İncel. Her continuous support, deep understanding, and constructive feedbacks have made these last five years both productive and rewarding. She consistently highlighted the positives, provided guidance through challenges, and helped me regain my motivation whenever I felt uncertain. As a new mom embarking on the demanding path of a PhD, I sometimes doubted whether I could manage it all. However, she never had any preconceived notions and always believed in me. This belief held special meaning for me and motivated me even more to push forward. Every conversation with her left me feeling recharged and inspired. I will always be grateful for her constant belief in me and the opportunity to work alongside her. The impact she has had on both my academic and personal growth is something I will carry with me throughout my life.

I also want to express my sincere appreciation to the members of my thesis jury. Albert Ali Salah enriched my work with his visionary and out-of-the-box thinking. He highlighted areas I had not considered enough and bringing overlooked aspects into focus. I am also grateful to Berk Gökberk for his practical advice that streamlined the process and for stepping in as my primary supervisor when my initial supervisor went abroad, continuing as a co-supervisor. I also extend my thanks to İnci Meliha Baytaş for her insightful feedbacks that ensured my work was comprehensive, complete, and robust. Her deep learning courses were especially helpful in helping me gain a deeper understanding of the fundamentals of the subject. I am also thankful to Cem Ersoy and Emre Alptekin for their time reviewing my thesis, their thoughtful comments, and contributions during my thesis defense.

I thank Boğaziçi University, the Department of Computer Engineering, and all my professors for their invaluable teaching and support throughout these years. Additionally, I am deeply grateful to all the researchers and institutions that have made their course materials and educational resources freely available. Their efforts to share knowledge and make complex subjects more accessible have been immensely helpful to me during this process.

I thank all my friends who supported me throughout this journey. I am especially grateful to Özlem Özgen and Hilal Bayram for their constant presence and encouragement. I want to thank Fulya Koç for always being there whenever I needed and being such a wonderful sister to me. Additionally, I appreciate my friends Büşra Oğuzoğlu, Pınar Süngü, and Cansu Damla Yılmaz, whom I met during my PhD journey. I am glad our paths crossed. I also want to extend my thanks to Egemen İşgüder for his assistance with the thesis formatting and for helping me navigate the institute's processes whenever I needed support.

I want to sincerely thank to my best friend and husband, Gökhan Ağaç, whose constant love and encouragement make everything possible. His incredible sacrifices and continuous support throughout this demanding journey have been invaluable. He was always there for every aspect of caring for our child, managing the household, and dealing with the life's challenges. Without his presence and understanding, completing this process would have been impossible. I also want to express special thanks to our sweet son, Ahmet Arif. His joyful smiles, warm hugs, boundless energy, curiosity, and enthusiasm have been a constant source of happiness, inspiration, and motivation throughout this journey.

I feel extremely fortunate to have a family that stands by me through every dream and challenge, including this PhD journey. My mom, Perihan, is my best! Even when I was feeling lost, she reminded me of my strengths and helped me see the true purpose of life. Her support and prayers have been invaluable. I am grateful to my dad, İbrahim, for his unwavering interest in my work and invaluable advice. I owe thanks to my sister

Esra, whose kind words and encouragement have been a great support through tough times. I am so lucky to have her as my sister. I am grateful to my brother İsmail for all the help he has provided, including, but not limited to, discussing the field and keeping me updated. I thank them all for their constant support and for making this journey so much more meaningful.

Finally and most importantly, I am grateful to Allah for surrounding me with such wonderful people, presenting me with great opportunities, and giving me all the abilities I needed to make this PhD journey successful.

This thesis has been supported by Scientific and Technical Research Council of Turkey (TÜBİTAK), Department of Science Fellowships and Grant Programmes (BİDEB) under the 2211-A National PhD Scholarship Program, and by Bogazici University Research Fund (BAP) under the project number 19301p.

ABSTRACT

ENHANCING LIGHTWEIGHT MODELS FOR EFFICIENT SENSOR-BASED HUMAN ACTIVITY RECOGNITION

This thesis presents a comprehensive study on enhancing the performance of lightweight models for efficient sensor-based Human Activity Recognition (HAR) on resource-constrained devices, particularly in memory-constrained environments such as wearables with microcontrollers. By integrating convolutional block attention modules into the lightweight versions of the DeepConvLSTM model, originally proposed for sensor-based HAR, and the highly efficient SqueezeNet architecture, the study aims to boost recognition accuracy without increasing computational demands. The thesis examines the impact of channel and spatial attention mechanisms across various model sizes. Results demonstrate that attention-enhanced lightweight models can achieve performance levels comparable to larger, more resource-intensive models while maintaining minimal resource usage. Subsequently, combinations of knowledge distillation and attention mechanisms were applied to further improve model efficiency. It was found that attention-based distillation could significantly enhance the accuracy of lightweight models even in the absence of attention modules in the teacher model. The effectiveness of the developed models was then compared with model compression techniques such as quantization and pruning. Furthermore, this thesis provides the most comprehensive study to date on sensor-based HAR in resource-constrained environments, offering insights and strategies for the practical application of efficient and high-performance sensor-based HAR systems, thereby addressing a significant gap in the field.

ÖZET

VERİMLİ SENSÖR TABANLI İNSAN AKTİVİTESİ TANIMA İÇİN HAFİF MODELLERİN İYİLEŞTİRİLMESİ

Bu tez, özellikle mikrodenetleyicilere sahip giyilebilir cihazlar gibi bellek ve diğer kaynakların sınırlı olduğu ortamlarda, sensör tabanlı İnsan Aktivitesi Tanıma (İAT) için verimli ve hafif modeller geliştirmeye yönelik kapsamlı bir çalışma sunmaktadır. Sensör tabanlı İAT için önerilmiş olan DeepConvLSTM modelinin ve yüksek verimli SqueezeNet mimarisinin hafif versiyonlarına konvolüsyonel blok dikkat modülleri entegre edilerek, hesaplama gereksinimleri artırılmadan tanıma doğruluğunu artırma hedeflenmektedir. Çalışma, kanal ve mekansal dikkat mekanizmalarının farklı model büyüklüklerindeki etkilerini incelemektedir. Sonuçlar, dikkat mekanizmalı hafif modellerin, daha büyük ve daha fazla kaynak gerektiren modellerle karşılaştırılabilir performans seviyelerine minimum kaynak kullanımıyla ulaşabileceğini göstermektedir. Sonrasında, modelin verimliliğini daha da artırmak amacıyla bilgi damıtma ve dikkat mekanizmalarının çeşitli birleşimleri kullanılmıştır. Dikkat tabanlı damıtmanın, öğretici modelde dikkat modülleri bulunmasa bile hafif modellerin doğruluğunu önemli ölçüde artırabileceği gösterilmektedir. Daha sonra, geliştirilmiş modellerin etkinliği, budama ve sayısallaştırma gibi model sıkıştırma teknikleriyle karşılaştırılmıştır. Ayrıca, bu tez, kaynak sınırlı ortamlarda sensör tabanlı İAT konusunda bugüne kadar yapılmış en kapsamlı çalışmayı sunarak, verimli ve yüksek performanslı sensör tabanlı İAT sistemlerinin pratik uygulaması için içgörüler ve stratejiler sunarak alandaki önemli bir boşluğu doldurmaktadır.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . iii
ABSTRACT . . . vi
ÖZET . . . vii
LIST OF FIGURES . . . xi
LIST OF TABLES . . . xiv
LIST OF SYMBOLS . . . xv
LIST OF ACRONYMS/ABBREVIATIONS . . . xvi
1. INTRODUCTION . . . 1
1.1. Research Overview and Questions . . . 6
1.2. Contributions and Thesis Outline . . . 7
2. LITERATURE REVIEW . . . 12
2.1. Motion Sensors . . . 12
2.2. Attention in Sensor-Based Human Activity Recognition . . . 14
2.3. Resource-Efficiency in Deep Learning Models . . . 16
2.4. Resource-Efficiency in Sensor-Based Human Activity Recognition . . . 19
2.4.1. Lightweight Model Design . . . 21
2.4.2. Knowledge Distillation . . . 22
2.4.3. Model Pruning and Parameter Quantization . . . 24
2.4.3.1. TensorFlow Lite Issues, Limitations, and Constraints . 31
3. DEEP LEARNING MODELS AND DATASETS . . . 33
3.1. Baseline Models . . . 33
3.1.1. Deep Convolutional and LSTM Recurrent Neural Network . . . 33
3.1.2. SqueezeNet . . . 34
3.2. Datasets . . . 36
3.2.1. Opportunity . . . 37
3.2.2. Sensors . . . 38
3.2.3. Wisdm . . . 39
3.2.4. Pamap2 . . . 39

3.3. Implementation Details and Evaluation Metrics 40
4. INTEGRATING CONVOLUTIONAL BLOCK ATTENTION MODULES IN DEEP LEARNING FOR SENSOR-BASED HAR 43
4.1. Convolutional Block Attention Module 43
4.1.1. Channel Attention Module 44
4.1.2. Spatial Attention Module 45
4.2. Experimental Results and Analysis of Attention Modules 48
4.2.1. Evaluating the Impact of Channel Attention 51
4.2.2. Evaluating the Impact of Spatial Attention 54
4.2.3. Evaluating the Impact of Channel-Spatial Attention 55
4.3. Conclusion and Insights 58
5. ACHIEVING RESOURCE-EFFICIENT HAR WITH LIGHTWEIGHT DEEP MODELS ENHANCED BY ATTENTION 60
5.1. Designing Resource-Efficient Models Using Base Models 60
5.2. Impact of Attention Modules on Different Model Sizes 62
5.3. Activity-Based Impact 67
5.4. Conclusion and Insights 71
6. BOOSTING THE PERFORMANCE OF LIGHTWEIGHT MODELS WITH ATTENTION MECHANISM AND KNOWLEDGE DISTILLATION 73
6.1. Knowledge Distillation 74
6.2. Combination of KD and Attention 75
6.2.1. Response and Attention-based KD 76
6.2.2. Response-based KD with Attention Applied to the LM 77
6.3. Impact of KD versus Attention over the Lightweight Model 77
6.4. Impact of Combined KD and Attention over the Lightweight Model . . 81
6.5. Conclusion and Insights 88
7. COMPARING THE PERFORMANCE OF BOOSTED LIGHTWEIGHT MODELS WITH COMPRESSED MODELS USING QUANTIZATION AND PRUNING 90
7.1. Model Compression with TensorFlow Lite 90
7.2. Parameter Quantization 91

7.3. Model Pruning 95
7.4. Recognition Success and Resource Consumption Comparison of Compressed Original Models Versus Boosted LM 102
7.5. Conclusion and Insights 105
8. CONCLUSION 107
8.1. Findings 107
8.2. Insights Gained and Recommendations 109
8.3. Limitations and Future Directions 111
REFERENCES 114

LIST OF FIGURES

Figure 1.1. Comparison of memory and storage capacities. 2

Figure 1.2. Thesis outline, including chapters, keywords, and RQs. 10

Figure 2.1. Simple illustration of three compression techniques. 18

Figure 3.1. Positions of wearable sensors. 37

Figure 3.2. Two-dimensional t-SNE visualizations of HAR datasets. 38

Figure 4.1. Application of channel attention and spatial attention mechanisms individually and in combination using CBAM. 46

Figure 4.2. DeepConvLSTM architecture with the CBAM. 47

Figure 4.3. SqueezeNet architecture with the CBAM. 48

Figure 4.4. Impact of channel attention on recognition performance. 52

Figure 4.5. Impact of spatial attention on recognition performance. 54

Figure 4.6. Impact of channel-spatial attention on recognition performance. . 56

Figure 5.1. Performance results using base models and attention models. . . . 63

Figure 5.2. Activity-based performance of LM, LM-Att and MM. 66

Figure 5.3. Confusion matrices. 69

Figure 6.1. Response-based KD system. 74

Figure 6.2. Response and attention-based KD (RAB-KD). 76

Figure 6.3. Response-based KD with attention applied to the student LM (RB-KD-Att). 77

Figure 6.4. Changes in recognition performance for individual use of attention module (LM-Att) and response-based distilled LM (RB-KD) over baseline LM. 78

Figure 6.5. Changes in resource usage for individual use of attention module (LM-Att) and response-based distilled LM (RB-KD) over baseline LM. 78

Figure 6.6. Changes in recognition performance for response and attention-based distillation on LM (RAB-KD) over baseline LM. 82

Figure 6.7. Changes in resource usage for response and attention-based distillation on LM (RAB-KD) over baseline LM. 82

Figure 6.8. Changes in recognition performance for response-based distillation on LM with attention (RB-KD-Att) over baseline LM. 85

Figure 6.9. Changes in resource usage for response-based distillation on LM with attention (RB-KD-Att) over baseline LM. 85

Figure 6.10. Ball chart reporting the recognition performance vs. resource consumption of LM, boosted LMs and MM. 87

Figure 7.1. Validation accuracy of quantized models. 93

Figure 7.2. Inference time of quantized models. 93

Figure 7.3. Model size of quantized models. 93

Figure 7.4. Validation accuracy of CP method across different pruning rates. . 96

Figure 7.5. Inference time of CP method across different pruning rates. 96

Figure 7.6. Model size of CP method across different pruning rates. 96

Figure 7.7. Validation accuracy of pruned models with pruning rate 60%. . . . 101

Figure 7.8. Inference time of pruned models with pruning rate 60%. 101

Figure 7.9. Model size of pruned models with pruning rate 60%. 101

Figure 7.10. Inference time comparison of original models, boosted LM models and compressed original models. 103

Figure 7.11. Model size comparison of original models, boosted LM models and compressed original models. 103

Figure 7.12. Validation accuracy comparison of original models, boosted LM models and compressed original models. 104

Figure 8.1. Flowchart for lightweight model selection based on insights gained and recommendations. 110

LIST OF TABLES

Table 1.1. SRAM and Flash memory capacities of various MCU/IoT. 1

Table 2.1. Studies using attention mechanism and/or model compression for sensor-based HAR. 28

Table 2.2. Performance results of studies using attention mechanism and/or model compression for sensor-based HAR. NA: Not available. . . . 30

Table 3.1. Details about datasets. 36

Table 4.1. Abbreviations used for attention settings in the experiments. . . . 49

Table 4.2. Additional params for DeepConvLSTM in attention experiment. . 49

Table 4.3. Additional params for SqueezeNet in attention experiment. 50

Table 4.4. Performance of base models and best attention models. 57

Table 5.1. Structure of the DeepConvLSTM models at different sizes. 61

Table 5.2. Structure of the SqueezeNet models at different sizes. 62

Table 5.3. Comparison of datasets using LM with attention and LM, MM, and OM without attention. 64

LIST OF SYMBOLS

L_{att}	Attention loss
L_{ca}	Channel attention loss
L_{dist}	Distillation loss
L_{sa}	Spatial attention loss
L_{stud}	Cross-entropy loss of student model
$Loss$	Overall loss
$\mathbb{R}$	Real numbers
r	Reduction ratio
α	Weighting factor for combining losses
β	Weight of attention loss
σ	Sigmoid activation
τ	Temperature in knowledge distillation
$\otimes$	Element-wise multiplication
$\oplus$	Element-wise summation

LIST OF ACRONYMS/ABBREVIATIONS

1D	One Dimensional
2D	Two Dimensional
3D	Three Dimensional
CA	Channel Attention
CBAM	Convolutional Block Attention Module
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CP	Constant Pruning
CSA	Channel and Spatial Attention
DeepConvLSTM	Deep Convolutional and LSTM Recurrent Neural Network
DRQ	Dynamic Range Quantization
FLOPs	Number of Floating-point Operations
FQ	Float16 Quantization
GRU	Gated Recurrent Unit
HAR	Human Activity Recognition
IMU	Inertial Measurement Unit
KD	Knowledge Distillation
LM-Att	Lightweight Model with Attention
LM	Lightweight Model
LSTM	Long Short-Term Memory
MCU	Microcontroller Unit
MLP	Multilayer Perceptron
MM-Att	Moderate-size Model with Attention
MM	Moderate-size Model
NA	Not Available
NAS	Neural Architecture Search
OM-Att	Original Model with Attention
OM	Original Model

params	Number of Trainable Parameters
PDP	Polynomial Decay Pruning
RAB-KD	Response and Attention-based Distilled Lightweight Student Model
RB-KD-Att	Response-based Distillation with Attention Applied to Lightweight Student Model
RB-KD	Response-based Distilled Lightweight Student Model
RNN	Recurrent Neural Network
RQ	Research Question
SA	Spatial Attention

1. INTRODUCTION

With technological advancements, both the variety and usage rates of wearable devices are increasing rapidly. These devices are equipped with multiple sensors, especially accelerometers and gyroscopes, and emerging as an ideal platform for tracking users' body movements, offering both cost-effective and flexible solutions [1, 2]. Currently, developing sensor-based Human Activity Recognition (HAR) systems is an active research area [3]. Despite substantial advancements [4], the field continues to present significant challenges, such as privacy concerns, imbalanced datasets, the complexities of concurrent activity recognition, and the optimization of energy consumption for on-device implementations [5].

Table 1.1. SRAM and Flash memory capacities of various MCU/IoT.

MCU/IoT	**SRAM**	**Flash Memory**
Nordic Thingy:52 [6]	32KB	256KB
STM32F407VG [7]	192KB	1MB
STM32F746NG [8]	320KB	1MB
Arduino Zero [9]	32KB	256KB
Arduino Due [10]	96KB	512KB
Arduino Nano 33 BLE [11]	256KB	1MB
Raspberry Pi Pico W [12]	264KB	2MB

Microcontroller Units (MCUs), the key component of wearable IoT devices, are becoming increasingly popular, especially as edge devices shift from cloud-dependent operations to more localized ones. Considering hardware options, MCUs are considered as the most attractive choice for wearable devices due to their low power consumption, cost, size, latency, and flexibility [13]. The widespread adoption of billions of MCUs today underscores their significance in modern wearable IoT technology [14, 15]. A major challenge faced by these edge devices, such as MCUs in wearable devices, is their limited resources, when compared to more powerful mobile and cloud platforms. While mobile devices also suffer from resource limitations like energy and latency, MCUs add further severe memory-related resource constraints [1]. This can be noted in

the Flash and SRAM capacities of some popular MCU devices. For example, Hexiwear smartwatch from MikroElektronika contains a low-power MCU that includes an ARM Cortex-M4 central processing unit (CPU) by NXP, equipped with 1MB of Flash and 256KB of SRAM [16]. In addition, Nordic Thingy:52 [6] is an IoT sensor kit with 32KB SRAM and 256KB Flash, and can be used as a wearable device to collect sensor data [17]. Besides, the Arduino Nano 33 BLE offers 256KB of SRAM and 1MB of Flash, and the STM32F746ZG provides 320KB of SRAM and 1MB of Flash, thus showing their limited memory resources (see Table 1.1). In contrast, most mobile devices are equipped with far greater storage and memory, for example, 4GB of RAM and 64GB of storage, as presented in Figure 1.1. These compact MCUs can be integrated with sensors to function as wearables or are already integrated into existing wearables, as in the example of the Hexiwear smartwatch. Besides, wearables are composed of a microprocessor, battery, and sensors, and for ease of carrying, these devices need to be small in size and light in weight, making compact processors like MCUs ideal. Consequently, this thesis focuses on addressing the unique challenges posed by tiny wearables, particularly those with limited memory capacities. Referred to hereafter as *target devices* or *wearables*, these devices underscore the flexibility and potential in modern technology applications.

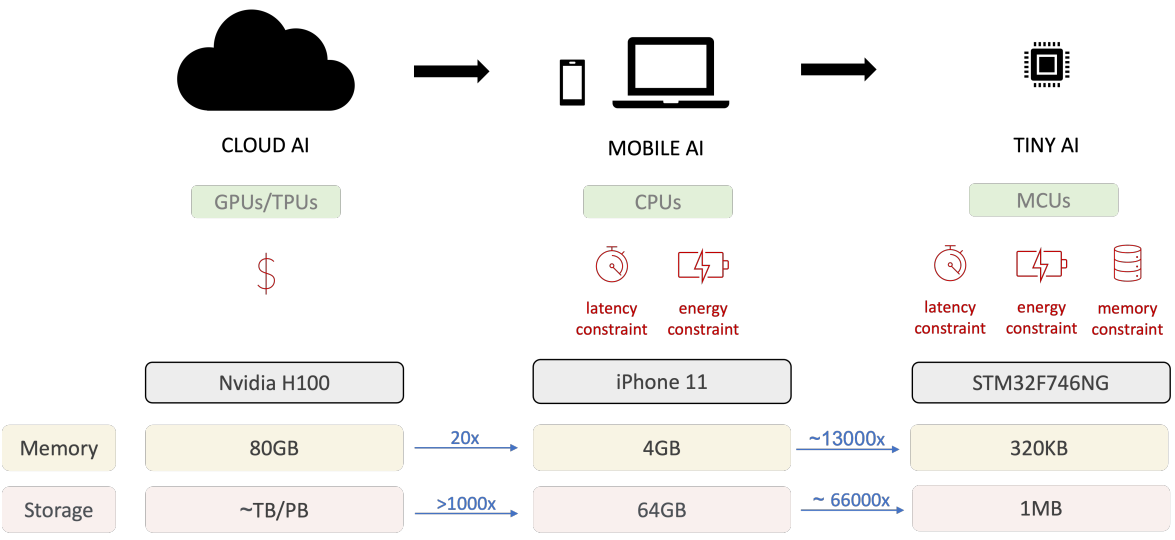


Figure 1.1. Comparison of memory and storage capacities.

In recent years, sensor-based HAR using deep learning techniques has proven to be highly effective in various fields [5], [18, 19], such as healthcare [20–22] and sports [23, 24]. However, implementing these deep learning models on devices with limited resources, like wearables, presents significant challenges due to deep models' substantial computational and memory demands. These constraints highlight the need for accurate and lightweight models that are low in complexity and, consequently, in resource consumption while capable of real-time and energy-efficient processing.

To tackle these challenges, the focus has shifted towards creating lightweight models that achieve high enough recognition accuracies. Numerous strategies have been proposed at the model, algorithm, and hardware levels to address the resource demands of deep learning models [25]. For instance, model pruning [26] reduces the complexity of model structures by eliminating unnecessary neurons or layers, thus decreasing the number of parameters [27, 28]. Similarly, quantization [29, 30] reduces the precision of model parameters to save resources [28], [31]. However, despite their frequent use, these techniques are not solutions that can be applied blindly [32]. The efficiency of pruning relies on the hardware, and it can require special hardware support, which raises problems with deployment. Quantization, while beneficial in reducing resource usage, can lead to a loss in accuracy if not managed correctly and is limited by compatibility issues with the target devices. Besides, although tools like TensorFlow Lite [33] support quantization and pruning, these methods only reduce resource consumption to a certain extent and might be insufficient for extremely resource-constrained devices, particularly due to their limited reduction in storage size [28], [34]. The issue is further complicated by the fact that edge devices have a wide variety of hardware and sensor modalities, which creates specific compatibility problems and hardware constraints that make such techniques challenging to apply.

Several studies have explored optimizing deep learning models for sensor-based HAR to reduce computational demands and improve deployment on resource-constrained devices. Contoli et al. [35] employed pruning to optimize deep models using the UCI-HAR dataset [36], achieving significant reductions in model size and computational

requirements, though facing challenges with TensorFlow Lite conversion and deployment on ESP32. A 1D Convolutional Neural Network (CNN) with quantization on a RISC-V MCU, maintaining high accuracy while reducing memory usage and energy consumption is implemented in [37]. Ghibellini et al. [38] used a 1D CNN on an Arduino Nano 33 BLE for industrial HAR, achieving 97% accuracy and substantial model size reduction through dynamic range quantization. In other studies [39] and [40], researchers focused on quantizing deep learning models and deploying them on mobile devices, demonstrating significant resource savings without compromising accuracy.

Despite these investigations, there remain significant gaps and areas for improvement in resource efficient, sensor-based HAR. To the best of our knowledge, there is one study that examines the use of both quantization and pruning techniques in sensor-based HAR, which was recently published [35]. This study, however, focuses on standard architectures (CNN, LSTM, CNN-LSTM) rather than state-of-the-art sensor-based HAR models like DeepConvLSTM [41] or well-known efficient architectures like SqueezeNet [42]. Besides, the study uses only the UCI-HAR dataset. The lack of generalizability is also a concern, as in most studies, the models are tested on a single dataset, limiting their robustness.

In Knowledge Distillation (KD), a smaller student model is trained to imitate a larger teacher model, and it is proven to reduce resource consumption while maintaining high accuracy results [43, 44]. Recent studies in sensor-based HAR have explored KD to enhance model performance while addressing resource constraints. For instance, progressive cross-modal KD models have been developed to improve accuracy using skeleton-to-sensor data [45]. Additionally, an ensemble learning technique has been used to enhance response-based KD with multi-channel data employing DeepConvLSTM for the teacher and CNN for the student [46]. Besides, advancements in KD now involve integrating attention mechanisms to enhance information transfer between teacher and student models. This technique, known as attention distillation, boosts student network performance by transferring attention maps from the teacher, offering improvements over traditional KD methods like response-based distillation. While

attention distillation has been successful in fields like computer vision, to the best of our knowledge, its application in sensor-based HAR remains unexplored.

Recent studies have highlighted the potential of attention mechanisms in improving the accuracy of deep neural networks by focusing on the most relevant features and discarding irrelevant ones. These mechanisms improve the model's ability to understand and interpret data, leading to better performance. These mechanisms gained popularity initially in natural language processing [47] and computer vision [48], and have been recently applied to sensor-based HAR domain. For example, an attention-based Long Short-Term Memory (LSTM) network [49] has demonstrated performance improvements by selectively focusing on specific sensor modalities and time segments. More sophisticated implementations often combine attention mechanisms with CNNs, such as dual attention modules [50] and self-attention frameworks [51], which enhance model performance by highlighting relevant features at different layers of the network.

However, a significant gap remains in investigating attention mechanisms for state-of-the-art and lightweight sensor-based HAR models in resource-constrained environments. Current research mainly focuses on enhancing model performance without carefully considering the computational overhead, which is crucial for real-time applications on edge devices. For instance, while dual attention modules [50] improve accuracy, they also increase complexity, making them unsuitable for resource-limited settings. Lightweight attention mechanisms, such as the Convolutional Block Attention Module (CBAM) [52], initially proposed for computer vision, offer a potential solution by improving performance with minimal additional computational cost but have not yet been explored for sensor-based HAR. Furthermore, the impact of attention mechanisms on lightweight models requires further investigation.

This thesis addresses the above-mentioned issues by (i) integrating the CBAM into two well-known deep architectures across four sensor-based HAR datasets: Opportunity, Sensors, Wisdm, and Pamap2, (ii) investigating the impact of attention mechanisms on different sizes of deep models, with a particular focus on lightweight deep

learning models to enhance recognition performance while maintaining computational efficiency, (iii) exploring response-based KD and incorporating attention mechanisms into KD, specifically through attention-based distillation, and (iv) utilizing both pruning and quantization techniques to optimize models for deployment on devices with limited resources, and comparing these with lightweight models enhanced by attention and distillation. By focusing on these areas, this thesis aims to advance the state-of-the-art in efficient deep learning techniques for sensor-based HAR, providing insights and recommendations for practical applications in real-world scenarios where resources are constrained.

1.1. Research Overview and Questions

This thesis aims to address the key question: *"How can we achieve a better balance between recognition accuracy and resource efficiency by enhancing lightweight sensor-based HAR models instead of relying on larger ones?"*. To address this question, we explore the impact of attention mechanisms on lightweight, or in other words resource-efficient, models and investigate how combining them with knowledge distillation can further enhance their performance. Attention mechanisms have the potential to improve the accuracy of deep neural networks by focusing on the most relevant features, which is particularly valuable in sensor-based HAR. Additionally, we apply pruning and quantization techniques on large models to compare our results with those of these well-known methods. There is a notable lack of research specifically addressing and comparing these techniques in sensor-based HAR, which underscores the need for this study.

We conduct our experiments on four sensor-based HAR datasets, namely Opportunity, Sensors, Wisdm, and Pamap2, using two deep architectures: DeepConvLSTM and SqueezeNet. To comprehensively address our key question, we focus on several sub-research questions (RQ) that guide our investigation, which are as follows:

- RQ1: What are the impacts of individual channel attention and spatial attention mechanisms on the performance of DeepConvLSTM and SqueezeNet models across different HAR datasets?
- RQ2: How do various parameters, such as reduction ratios and kernel sizes within the attention modules, influence the trade-off between recognition performance and resource consumption?
- RQ3: Can the combination of channel and spatial attention mechanisms provide significant performance improvements compared to using each mechanism individually?
- RQ4: What is the effect of incorporating attention modules into different sizes of DeepConvLSTM and SqueezeNet models on recognition accuracy and resource consumption?
- RQ5: How does the individual use of attention and response-based distillation techniques affect the performance of lightweight models?
- RQ6: To what extent does incorporating attention mechanisms into knowledge distillation, particularly through attention-based distillation, enhance the performance of lightweight HAR models?
- RQ7: What are the effects of incorporating attention mechanisms into lightweight student models when their teacher models do not have attention modules?
- RQ8: How do different quantization techniques affect the accuracy and resource efficiency of original models?
- RQ9: How do different pruning techniques impact the recognition performance, model size, and inference time of original models?
- RQ10: How do compressed original models compare to boosted lightweight models in terms of recognition success, model size, and inference time?

1.2. Contributions and Thesis Outline

In this thesis, we aim to enhance the efficiency of deep learning models for sensor-based HAR in resource-constrained environments. By focusing on this objective, we strive to advance the state-of-the-art in efficient deep learning techniques for sensor-

based HAR, thereby enabling their practical application in real-world scenarios where resources are limited as in the case of wearables. Consequently, the contributions of this work can be organized into the following categories:

- *Integrating attention module, CBAM [52], into two deep architectures across four sensor-based HAR datasets (Chapter 4):* Although the effect of using attention with some deep learning architectures has been investigated in related studies, we are interested in exploring how the already high-performing DeepConvLSTM [41] and resource-efficient SqueezeNet [42] models can benefit from channel, spatial, or combined attention methods of CBAM. To this end, we conducted an extensive set of experiments to examine the impact of different reduction ratios, kernel sizes, and the application of attention at various depths within the deep architecture on recognition performance across four datasets with multiple body positions. Our findings show that the combination of integrating attention boosted the recognition performance of models. Furthermore, we found that adding attention did not significantly increase the number of additional model parameters and FLOPs, highlighting the potential for these less complex models to be ported to wearables or other tiny devices with similar resource constraints. This integration effectively balances computational efficiency and enhanced feature extraction, establishing CBAM as a valuable improvement for sensor-based HAR.
- *Exploring the impact of attention mechanisms on different model sizes (Chapter 5):* There is a lack of research on applying attention mechanisms in models of varying sizes, particularly lightweight ones, to improve model performance within the sensor-based HAR field. To fill this gap, we performed experiments to assess how attention mechanisms affect different model sizes, including lightweight, moderate-size, and original versions of DeepConvLSTM and SqueezeNet architectures. The results revealed that attention mechanisms notably improved the performance of lightweight models. Besides, smaller models show the most significant enhancements. These findings indicated that adding attention mechanisms to lightweight models can boost their performance while keeping resource usage low. This makes attention-enhanced lightweight models more efficient alterna-

tives to larger models without substantially increasing computational demands.

- *Combining knowledge distillation with attention (Chapter 6):* We observed that attention mechanisms improve the performance of lightweight models, and it is known from the literature that knowledge distillation can also enhance small models. However, the potential of combining attention mechanisms and knowledge distillation to further improve the performance of sensor-based HAR systems has not been explored. To address this, we conducted experiments investigating the combined impact of attention mechanisms and knowledge distillation on lightweight models. Our results demonstrated that this combined approach significantly improves the recognition accuracy of lightweight models, often surpassing traditional response-based knowledge distillation methods. Even when the teacher model lacks attention mechanisms, the results are still highly favorable.
- *Optimizing sensor-based HAR models with pruning and quantization (Chapter 7):* Although techniques such as quantization and pruning are well-known for their ability to reduce computational demands, their application to sensor-based HAR models has not been thoroughly studied. Therefore, we conducted experiments to evaluate the impact of two quantization methods (float-16 and dynamic range quantization) and two pruning approaches (constant and polynomial decay pruning). Our results demonstrated that these techniques significantly reduce computational and memory requirements while maintaining high recognition accuracy. We then compared the recognition performance and resource consumption of compressed original models with boosted lightweight models. Our analysis highlighted that boosted lightweight models (using attention and KD) did not match the recognition success of original models, but they reduced the f1-score gaps from 3.94%–75.26% to 2.65%–30.27%. In one case, the boosted lightweight model nearly matched the original, with only a 0.5% difference. While not fully matching the original models' recognition success, the boosted lightweight models have a compelling trade-off between recognition accuracy, computational efficiency, and memory usage. For devices with severe memory limitations, boosted lightweight models are recommended, whereas model compression techniques like pruning and quantization are preferred otherwise.

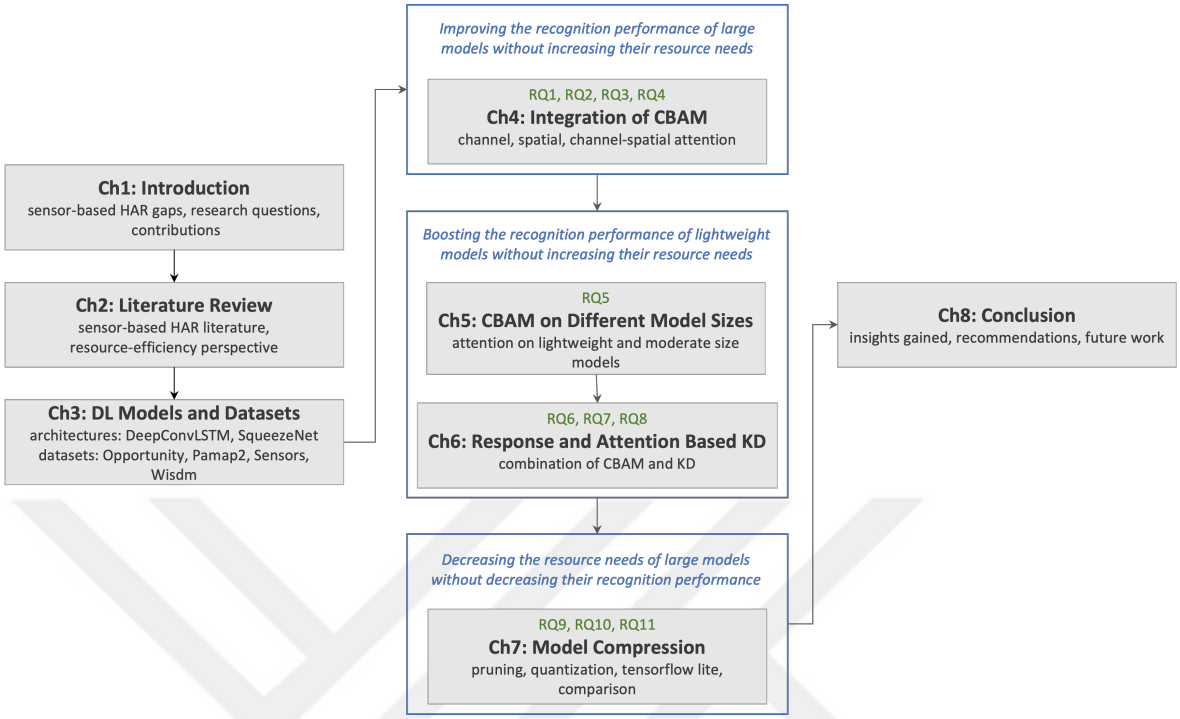


Figure 1.2. Thesis outline, including chapters, keywords, and RQs.

Figure 1.2 illustrates the organization of this thesis. Chapter 2 reviews the existing literature on deep learning for sensor-based HAR and resource-efficient deep learning techniques, identifying gaps in current research and setting the stage for the contributions of this thesis. Chapter 3 introduces the baseline models, DeepConvLSTM and SqueezeNet, and describes the datasets used in this thesis, outlining the implementation and training protocols, including the evaluation metrics. Chapter 4 investigates the integration of the CBAM into the deep learning models, evaluating the impact of attention mechanisms on model performance and resource efficiency. Chapter 5 focuses on designing and evaluating lightweight versions of the proposed models, exploring the impact of attention mechanisms on these smaller models and comparing their recognition performance and resource consumption with larger models. Chapter 6 examines the combination of attention mechanisms and knowledge distillation techniques to further boost the performance of lightweight models, presenting experimental results and insights into the effectiveness of these combined approaches. Chapter 7 compares the performance of boosted lightweight models with compressed models using quantization

and pruning techniques, providing a detailed analysis of the trade-offs between accuracy and resource consumption. Finally, Chapter 8 summarizes the main contributions and insights gained from this research, outlining the limitations of the current work and suggesting potential directions for future research to further advance the field.

2. LITERATURE REVIEW

In this section, we review recent advancements in the field of deep learning for sensor-based HAR, with a focus on studies that tackle the challenges of implementing models on devices with limited resources. We first discuss research on motion sensors embedded in resource-limited devices, such as wearables, and their impact on HAR. Then, we highlight research integrating attention mechanisms into deep learning models in order to improve performance by focusing on the most relevant parts of the input data. Additionally, we examine studies on model compression techniques such as quantization, pruning, and KD, which aim to reduce model size and computational load, making them more suitable for resource-constrained environments. We also discuss research on efficient and lightweight models designed to minimize computational demands while maintaining high recognition success. Furthermore, we review research that relies on specific datasets and discuss how this reliance impacts the generalizability of the created HAR models. By exploring these various approaches, we aim to identify potential areas for improvement and future research directions, with the goal of developing more efficient and generalizable HAR models suitable for deployment in practical, resource-constrained environments.

2.1. Motion Sensors

Inertial measurement unit (IMU) sensors, also known as inertial sensors, are extensively used in HAR due to their direct relation to body movements. These low-cost sensors are commonly embedded in commercial products such as smartphones, smart glasses, and smartwatches [53]. IMUs have different modalities, including accelerometers, gyroscopes, and magnetometers, all operating in three dimensions.

The accelerometer measures acceleration along the x, y, and z axes, with units in m/s^2. The gyroscope measures rotational motion in rad/s. Combining accelerometer and gyroscope data enhances the robustness of activity information, as they comple-

ment each other in understanding movement dynamics. For instance, [54] demonstrates a HAR system that combines accelerometer and gyroscope data to better capture user behavior changes, such as transitions between sitting and standing. However, using only accelerometer data from a smartwatch can effectively recognize activities like sitting, standing, and walking [55]. The gyroscope is particularly crucial for detecting specific movements such as food intake [56], especially when the sensor is placed on the wrist. However, its effectiveness diminishes when placed on the chest or leg for activities like smoking detection.

The magnetometer, or geomagnetic field sensor, measures changes in the Earth's magnetic field, with units in μT. When combined with accelerometers, magnetometers can provide promising results [57]. Sensor positioning on the body significantly impacts performance, and some studies emphasize using magnetometers to achieve sensor position invariance [58, 59]. A survey [60] shows that among HAR studies using inertial sensors, accelerometers are used in 149 research papers, gyroscopes in 83 alongside accelerometers, and magnetometers in 27 with accelerometers. This highlights that accelerometers are the primary choice among inertial sensors, often complemented by gyroscopes and magnetometers, respectively. While this thesis focuses on sensor-based HAR using IMUs, it is noteworthy that studies utilizing vision-based [61] and WiFi-based [62] signals for HAR also exist in the literature. These methods, however, fall outside the scope of this thesis.

The widespread adoption of wearable devices equipped with various sensors drives the popularity of these technologies. While some research uses single body-location sensors [36], [63, 64], a single sensor may only capture local movement patterns, reducing recognition accuracy for complex activities like rowing. Therefore, using multiple sensors at different body locations can better capture target activities, with each sensor providing specific information based on its placement [65, 66]. For instance, leg movements are more prominent during cycling, and wrist movements are significant during eating. Fusing data from different body locations enhances both accuracy and reliability [67]. Besides, nowadays, research focus has shifted to automated feature

extraction techniques rather than exploring manually extracted features [68]. Thanks to its multi-layered structure, deep learning allows automatic feature extraction in a wide range of features, from low-level to high-level. It increases the accuracy and the robustness of the models [18], and it is very popular in various domains such as sensor-based HAR, which is the focus of this thesis [69–75]. However, collecting and processing data from multiple sensors simultaneously increases computational demands, especially for complex tasks, thereby highlighting the importance of efficient resource management. Additionally, deep learning models, while improving accuracy and robustness, further amplify resource consumption due to their computationally intensive nature, underscoring the need for optimized algorithms and hardware.

2.2. Attention in Sensor-Based Human Activity Recognition

Attention mechanisms, inspired by the human ability to focus on relevant information while disregarding irrelevant ones, have revolutionized deep learning by allowing models to weigh different parts of the input data. Initially popularized in natural language processing [47] and computer vision [48], attention mechanisms have now found applications across various domains [76].

In recent years, attention mechanisms have been integrated into sensor-based HAR to improve the performance of deep learning models, with growing interest and evidence from studies such as [50,51], [77–79]. These mechanisms help models focus on the most relevant parts of the input data, which is particularly useful in HAR, where data from various sensors and body positions can vary in significance. For example, [49] introduced a continuous attention-based LSTM network that focuses on both different sensor modalities and time segments, boosting the recognition performance on two HAR datasets (Pamap2 [80] (from 75% to 90% f1-score) and Daphnet Gait [81] (from 67% to 84% f1-score)), and slight decreasing in Skoda dataset [82] (from 90% to 89% f1-score). Similarly, Ma et al. [83] combined spatial and temporal attention in a Gated Recurrent Unit (GRU) network. Proposed model, AttnSense, achieved 96%, 93% and 89% f1-score performance on HHAR [84], Skoda and Pamap2 dataset, respectively.

More advanced attention applications in sensor-based HAR involve combining attention mechanisms with CNNs. Studies such as [50] have implemented dual attention modules, leveraging both temporal and channel attention to enhance performance. The proposed model achieved 99%, 93%, 83% and 79% accuracy on the Wisdm [85], Pamap2, Opportunity [86], and Unimib-SHAR [87] datasets, respectively. It is also noted that the models require between 1.5M and 3.5M parameters. Considering that none of these datasets are balanced, with the Wisdm dataset being particularly highly imbalanced, the accuracy metric alone may be misleading. A self-attention framework is proposed in [51] to fuse heterogeneous and time-series information from various sensors. Another notable approach by [77] involved global position and modality attention modules, enhancing the model's ability to focus on relevant features at different layers of the network. Li et al. [78] explored a multi-branch attention-based grouped convolution network, applying attention to the convolutional layers of each branch separately, which, although effective, increased the computational burden significantly. To be more specific, the model achieved f1-scores of 99%, 97%, and 76% on the UT dataset [88], Pamap2, and Opportunity, respectively. In achieving these results, the proposed model required between 0.9M and 1.2M parameters and between 7.2M and 10.5M FLOPs. Furthermore, a cross-domain attention model with three parallel attention branches is presented in [89] presented. The study focuses on temporal-spatial, temporal-channel, and channel-spatial interactions using four public HAR datasets (Pamap2, Wisdm, UCI-HAR [90] and Unimib-SHAR). The proposed model achieved approximately 1% f1-score increase without significantly increasing the number of parameters. However, while the use of three branches resulted in improved recognition performance, it also came with increased computational cost.

Despite these advancements, there remains a significant gap in research focusing on the deployment of attention mechanisms in lightweight models suitable for resource-constrained environments. Current studies predominantly enhance model performance without considering the computational overhead, which is a critical factor for real-time applications on edge devices. For instance, the model proposed in [50] achieved higher accuracy with dual attention but at the cost of increased model parameters, making

it less feasible for deployment in resource-limited settings. Similarly, in [91], authors added attention to the LSTM layers of DeepConvLSTM, improving performance but not addressing the resource constraints adequately. While Tao et al. [92] and Wang et al. [93] integrated attention mechanisms into their models, the increased computational complexity and resource demands make them less suitable for lightweight applications. Besides, although studies such as [89], [92–94] examined attention modules across various architectures, none address the use of attention mechanisms in models of different sizes, particularly lightweight models, within the sensor-based HAR domain. Despite notable improvements, resource constraints still limit the feasibility of deploying deep sensor-based HAR models like DeepConvLSTM.

Since attention mechanisms help models prioritize important features, exploring their application in deep models with limited feature extraction capacity is essential. In this thesis, we aim to address this gap by evaluating the effectiveness of attention mechanisms in lightweight models, particularly within resource-constrained environments. As a starting point, we examined how attention mechanisms impact both recognition performance and resource usage of two current state-of-the-art models, one proposed for sensor-based HAR (DeepConvLSTM) and one known for its resource-efficient design (SqueezeNet). Subsequently, we investigated the effects of attention mechanisms across different model sizes (lightweight, moderate-size and original). By focusing on the practicality and efficiency of these mechanisms, we aim to improve recognition performance without significantly increasing computational costs. As a result, these boosted models are more accessible for deployment on wearable devices, paving the way for practical and efficient real-time activity recognition systems.

2.3. Resource-Efficiency in Deep Learning Models

Developing and optimizing efficient deep learning models for mobile devices is still relatively new [32], and it is an even newer field for more compact devices such as wearables. When creating a model that aims to fit within resource constraints, two primary approaches might be considered. One approach is to start with a deep model

that is already resource-efficient but may not perform sufficiently well and then try to enhance its performance without significantly increasing resource consumption. Another approach involves taking a high-performing, resource-intensive model and trying to reduce its resource needs as much as possible without significantly compromising performance.

The first approach is often referred to as lightweight model design, which involves creating models that are inherently small and efficient. Specifically, we take an existing deep model and redesign it to use fewer parameters and require less computational power [95]. Examples of this approach include models like MobileNets [96], ShuffleNets [97], and SqueezeNets [42], which are recognized as three representatives manually designed efficient CNN architectures [98]. MobileNets, for example, use depthwise separable convolutions to significantly reduce the number of parameters and computational cost while maintaining accuracy [98, 99]. ShuffleNets further optimize efficiency by utilizing pointwise group convolution and channel shuffle, ensuring a balance between computational cost and model performance [98–100]. SqueezeNet achieves AlexNet-level [101] accuracy with 50x fewer parameters, making it suitable for deployment on devices with limited resources [98–100]. Alternatively, we can directly create a model with a small number of parameters by considering the constraints of our device and the type of problem we are dealing with (e.g., image or time series), such as a CNN with just a few layers for image classification.

Moreover, advancements in neural architecture search (NAS) [102] have further contributed to resource efficiency in deep learning. NAS automates the design of neural network architectures to find optimal models that balance performance and resource efficiency. This technique has led to the development of models which integrate efficiency directly into the architecture design process [98, 99]. However, NAS requires significant computational resources and time, demanding powerful hardware and prolonged execution time. Studies highlight the intensive computational demands of NAS during the training phase of each search [103], and NAS often involves truncated training that limits the training duration or resources allocated to each architecture, leading

to incomplete learning, suboptimal performance assessment and biases that impact the accuracy of comparisons between different architectures [104]. Therefore, in this thesis, we focused on more feasible and practical alternatives that are easier to implement. These methods include manually decreasing the model size (e.g., by reducing the number of filters), applying pruning, quantization, and utilizing knowledge distillation rather than employing NAS directly. Despite advancements and efforts in resource-efficient model design, running them on tiny devices remains challenging due to limited hardware resources. For example, the Arduino Nano 33 BLE has 256KB SRAM and 1MB Flash, the STM32F746 offers 320KB SRAM and 1MB Flash and ultra-low-power devices like the RISC-V MCU with 64KB SRAM and 256KB Flash. However, these resources are insufficient for models like SqueezeNet, which have model sizes and Flash memory needs measured in MBs.

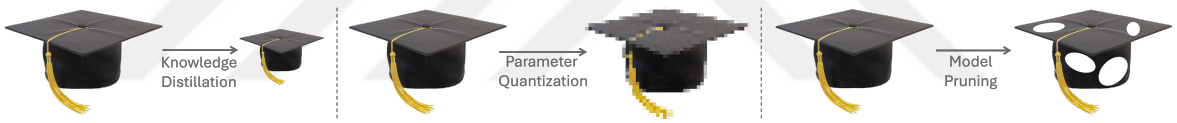


Figure 2.1. Simple illustration of three compression techniques.

The second approach can be achieved through more automatized techniques such as knowledge distillation, parameter quantization, and model pruning. Knowledge distillation involves transferring knowledge from a large, complex model (teacher) to a smaller, simpler model (student), enabling the smaller model to achieve improved or comparable performance with significantly reduced resource requirements [98]. Parameter quantization reduces the precision of model parameters, such as converting weights from 32-bit floating point to 16-bit, thereby decreasing memory usage and computational overhead without major loss in accuracy [98], [100]. Model pruning, on the other hand, eliminates redundant parameters and neurons, reducing the model size and inference time while retaining essential features necessary for maintaining performance [98]. Figure 2.1, inspired from [32], presents a very simple illustration of the three compression techniques.

2.4. Resource-Efficiency in Sensor-Based Human Activity Recognition

Although innovations and advantages brought by deep learning algorithms are desired to be used in the field of HAR to increase recognition performance, minimizing resource consumption, specifically battery and memory usage, and CPU time, is a challenging issue in wearables. These devices are especially limited in terms of battery and memory, and even running the algorithms on these devices in the inference mode is a problem that requires extra effort. Note that sending the sensor data to a central device is also an option. However, this approach causes privacy concerns and incurs additional costs for wireless data transmission, which are outside the scope of our thesis.

In multi-sensor-based HAR studies, multiple sensors produce large amounts of data, which makes the resource consumption problem more severe. For example, the DSADS dataset has 5 IMUs with 3-axis 3 motion sensors, which results in 45 signals for each time step. Therefore, researchers started to look for solutions to the challenging computation capacity and power resources requirements of deep learning algorithms for mobile and wearable technologies [105]. This problem is considered among future directions and has gained attention in recent surveys about IoT devices, in general, [53], [106, 107] and also about sensor-based HAR [5], [105], [108, 109] more specifically.

Resource consumption can be measured directly using metrics such as CPU time, memory usage, and energy consumption. However, since CPU and energy consumption are device-dependent, the number of parameters in the model is often used in studies to give an idea of resource consumption. Various studies [50], [77], [110, 111] analyzed resource consumption, particularly in terms of total model parameters, to demonstrate the effectiveness of a proposed model or to show that the suggested model does not significantly increase resource needs in the wearables/mobiles domain. Xu et al. [112] proposed a deep learning approach that combines a CNN and a Recurrent Neural Network (RNN). In the CNN part, an inception neural network is used to extract multi-dimensional spatial features by using various kernel-based convolution layers. Then, the extracted features are used in RNN for classification. They tested

their approach on three publicly available HAR datasets (Opportunity, Pamap2 and Smartphone dataset [113]) and compared their performances with CNN [114] and DeepConvLSTM [41]. It is stated that the proposed system performs better (around 3% f1-score over DeepConvLSTM) in terms of recognition. The model is also imported into an embedded device for efficiency analysis, and the running time of the model on test data is presented.

Similarly, Yao et al. [115] also combined CNNs and RNNs and tested their performance on car tracking, HAR and user identification areas. They imported the models on Nexus 5 with Qualcomm Snapdragon 800 SoC and Intel Edison Compute Module. Then, they investigated the performance of their system in terms of accuracy, latency and consumed energy. The study is then extended in [51] where an attention mechanism is included in the system to balance the effect of multiple sensors and increase the recognition performance.

Peppas et al. [116] proposed a real-time HAR system. They create the classification model using CNN architecture in which accelerometer sensor data is used as input, and 1D convolutional and max-pooling layers are used for feature extraction. Then, the output is combined with statistical features and fed to the fully connected layers for classification. The trained models are exported to a mobile device using the TensorFlow Lite framework. They investigated the performance of the exported system in terms of the number of model parameters, throughput, and model size and also compared it with [117]. They used the Wisdm and Actitracker Datasets [118] in their evaluations. They observed that the proposed system has approximately 7.3 times fewer parameters and 5x to 8x smaller model size compared to the reference model proposed in [117].

Although resource consumption analysis is implemented in some studies, a real effort to decrease resource consumption of an existing model is limited. The issue of minimizing the computational needs of deep neural networks has been largely disregarded by the research community even though the wearables have scarce memory and

computational resources [119]. In the following subsections, we explore sensor-based HAR studies aimed at minimizing the resource consumption of existing large models and improving the recognition performance of existing small models without increasing their resource consumption.

2.4.1. Lightweight Model Design

Zhou et al. [120] addressed computational and data constraints by developing a model significantly smaller than the state-of-the-art models such as DeepConvLSTM. The model emphasizes multi-modal sensor data processing and temporal information extraction while maintaining competitive performance on benchmark datasets. Despite its notable success across Pamap2, DSADS [121], and Daphnet Gait datasets, the reduction in model size sometimes leads to a decrease in recognition accuracy for complex activities or datasets with higher variability, as in the case of the Opportunity dataset. Zou et al. [122] pointed out that in the model proposed in [120], model size is reduced primarily by decreasing the number of hidden layers, which might not lower computational costs since it has sub-networks embedded with attention mechanisms. Bock et al. [111] removed one LSTM layer from the DeepConvLSTM architecture, validating this modification on five publicly available HAR datasets to demonstrate the potential of this simplified architecture for resource-constrained environments.

Zou et al. [122] enhanced spatiotemporal feature fusion while keeping the model lightweight, targeting real-time applications on wearable devices. They experimented on four datasets: Pamap2, DSADS, RealDisp [123] and USC-HAD [124]. The study presented an innovative spatiotemporal fusion network that leverages the complementary properties of spatial and temporal features to achieve high recognition accuracy with minimal computational overhead. Ankita et al. [125] introduced an optimized model specifically designed for smartphone sensors, optimizing performance and computational requirements through techniques such as parameter reduction and efficient layer design. However, they experimented on only one HAR dataset (UCI-HAR) and the focus on smartphone sensors may limit the model's applicability to other types of

wearable devices.

Coelho et al. [126] proposed a system designed for efficient operation on wearable devices, utilizing simplified architectures and optimization techniques to maintain high accuracy. This framework combined feature extraction with lightweight convolutional neural networks and feature selection to reduce computational overhead and enhance robustness. The multi-stage pipeline processed raw sensor data, extracted and selected relevant features, and used a hybrid classification approach that integrates traditional machine learning with deep learning. Extensive experiments showed that the framework achieves high accuracy, low latency, and low power consumption across various benchmarks, making it suitable for resource-constrained wearable devices. However, the framework's evaluation was limited to the MHEALTH dataset [127], which may not fully capture the diversity of real-world activities and environments, potentially limiting its generalizability.

2.4.2. Knowledge Distillation

Knowledge distillation is employed as a fundamental method to reduce resource consumption of deep models [25]. It includes three sub-methods: response-based, feature-based, and relation-based knowledge distillation [44]. Response-based KD focuses on the final output of the teacher model, where the student mimics the predicted probabilities. Feature-based KD uses the internal features learned by the teacher to guide the student. Relation-based KD captures and preserves relationships between different data samples in the teacher model. Besides, response-based is favoured for its effectiveness in model compression and wide application across tasks and domains [44].

Recent studies have investigated KD for HAR. For example, Chen et al. [45] presented a progressive cross-modal (skeleton-to-sensor) KD model that enhances performance while addressing communication delays and computational power constraints within a cloud-edge platform. They developed a student model that only requires time-series data, such as accelerometer data from a smartwatch during inference, and

multiple teacher models based on human skeleton sequences. Deng et al. [46] improved teacher's performance using ensemble learning and enhanced response-based KD with multi-channel data, employing DeepConvLSTM for the teacher and CNN for the student. They conducted their experiments on the Opportunity and Pamap2 datasets, achieving better recognition performance with improved resource efficiency. Additionally, Mardanpour et al. [128] proposed a feature and response-based KD system for sensor-based HAR, using tucker decomposition to create more informative feature maps and thus improve KD performance. They evaluated the proposed system on the RealDisp [123] dataset and found that the student model could achieve better accuracy with significantly reduced complexity.

Ni et al. [129] introduced a progressive cross-modal knowledge distillation method that uses skeleton data to train accelerometer-based HAR models, achieving notable improvements in accuracy. This technique involved constructing multi-teacher models and employing a progressive learning scheme to reduce the performance gap between teacher and student models. Similarly, Yuan et al. [130], and Ni et al. [131] explored multi-modal KD frameworks that integrate visual and inertial sensor data, thereby enhancing the robustness of HAR systems. It is important to note that multi-modal approaches, while beneficial, need to be carefully designed to avoid excessive computational overhead. Jeon et al. [132] investigated the role of data augmentation strategies in KD for wearable sensor data, highlighting how augmented data can help student models learn more effectively from teacher models. Ullah et al. [133] proposed a 3D CNN-based KD framework for HAR, demonstrating the potential of deep convolutional networks in distilling knowledge from high-dimensional data.

Further advancements in KD involve the integration of attention mechanisms, which enhance the information transfer between teacher and student models [48], [134]. This technique, known as attention distillation, significantly improves the classification performance of the student network by transferring attention maps from the teacher network, and it is recognized as an enhancement over traditional KD methods like response-based distillation [135]. Although attention distillation has been used in other

domains, such as computer vision, its application in sensor-based HAR remains an open question.

In a study from the computer vision field, [136], addressed a student-teacher model for knowledge distillation, testing it on benchmark datasets Cifar-100 [137] and ImageNet [138] with ResNet [139] networks. They achieved state-of-the-art results on one dataset and found that the student outperformed the teacher on another. Their use of an attention-based channel-wise distillation loss was intended to capture the teacher's attention in the student. However, the potential for utilizing attention mechanisms to boost recognition performance while reducing resource consumption in sensor-based HAR has yet to be investigated.

2.4.3. Model Pruning and Parameter Quantization

Pruning involves reducing the number of parameters in a deep-learning model by eliminating unimportant weights. This technique helps to decrease model size and computational load, making it suitable for deployment on resource-constrained devices. Contoli et al. [35] investigated pruning and qunantization to optimize deep learning models for HAR by using CNN, LSTM and the combination of CNN-LSTM architectures. The experiments were conducted using only the UCI-HAR dataset. The study demonstrated significant reductions in model size and computational requirements without substantial loss in accuracy. However, the study faced several issues that limited certain experiments. A significant challenge was the limited support for TensorFlow Lite operators, which caused errors during the conversion process. Specifically, the built-in operators supported by TensorFlow Lite are limited to a subset of those available in TensorFlow, necessitating the enabling of specific TensorFlow operators to ensure successful model conversion. Additionally, the deployment on the ESP32 device encountered errors, particularly when using the pruning technique in combination with other quantization techniques. These errors prevented the authors from performing a complete performance analysis in cascading compression scenarios.

Daghero et al. [37] utilized a single-core RISC-V MCU to implement an efficient 1D CNN with quantization, optimizing for limited memory and computational resources. Experimental results on four HAR datasets (public Wisdm, Unimib-SHAR, UCI-HAPT [36] datasets and private WALK dataset) showed significant reductions in memory usage and energy consumption while maintaining high accuracy, making it suitable for real-time HAR on ultra-low-power devices. However, the absence of on-device training support for MCUs could limit the model's adaptability to new data, potentially reducing accuracy over time. Another limitation was that the study only considers integer quantization formats up to 8-bit, as many low-power MCUs, including the one used in the deployment experiments, lack a hardware floating-point unit. Bursa et al. [140] focused on transforming four deep architectures (LSTM, CNN, CONVLSTM2D, CNN-LSTM) to train models on mobile devices for sensor-based HAR purposes. Two commonly used benchmark datasets Wisdm and MobiAct [64] are used to show the success of the proposed system. They compared the optimized model (created using TensorFlow Lite) with the original model (created using TensorFlow) in terms of accuracy, model size, CPU, memory, and energy consumption. Results showed that the optimized model reduces resource consumption without sacrificing recognition performance, especially when using quantization methods. Ghibellini et al. [38] explored the implementation of HAR using low-power microcontrollers. Utilizing an Arduino Nano 33 BLE, which features a 32-bit ARM Cortex-M4 CPU and an inertial measurement unit, the study develops a 1D CNN for detecting critical movements such as falling and running in industrial environments. The CNN model achieved 97% accuracy and benefits from a dynamic range quantization technique, reducing the model size by 53%. One major issue was the reliance on integer quantization formats up to 8-bit, similar to [37], which can restrict precision and operational efficiency. Additionally, the small dataset used for training and evaluation may not adequately represent real-world scenarios, potentially affecting the model's generalizability.

Contoli et al. [141] investigated the impact of three techniques which are lite conversion, dynamic quantization, and full-integer quantization, on deep learning models deployed on low-power device ESP32 (4KB Flash memory and 400KB RAM). The

study, using UCI-HAR dataset, demonstrated that dynamic quantization and full-integer quantization can achieve significant energy savings of nearly 30% while maintaining a negligible drop in accuracy, making them suitable for real-time HAR applications on wearable devices. However, the study also identified several limitations. The most notable is the inherent trade-off between compression, accuracy, and energy efficiency, with full-integer quantization resulting in a 2-3% drop in accuracy. Additionally, some network configurations, particularly lite models of 1D CNN-LSTM, failed to fit within the memory constraints of the device, complicating deployment efforts. Similarly, Lin et al. [142] also highlighted that even after quantization, the model may still be too large to fit within the limited memory resources of tiny edge devices.

Zebin et al. [40] explored the development of a deep learning model for efficient deployment on edge devices. Utilizing a custom balanced dataset including accelerometer and gyroscope data from 20 subjects, the model aimed to classify five everyday activities: three walking-related, one sedentary, and one sleep activity. The architecture consisted of a four-layer CNN, which is quantized for optimized performance on edge devices. Implemented using TensorFlow Lite, the model was deployed on a Samsung A5 smartphone, demonstrating low power consumption and effective performance, with an accuracy of 96.4%. However, the study faced challenges, including the limitations of TensorFlow Lite, such as operator compatibility, the lack of on-device training support for microcontrollers, and managing imbalanced datasets during the training process. Bursa et al. [39] investigated the performance of various deep learning models targetting resource-constrained mobile devices for HAR. The study utilized three datasets: Wisdm, MobiAct, and OpenHAR [143], which include data from accelerometers and gyroscopes. Four deep learning architectures, CNN, LSTM, ConvLSTM2D, and CNN-LSTM were trained and optimized using TensorFlow Lite (with and without quantization), achieving average accuracy rates of 97%, 92%, and 95% on Wisdm, MobiAct, and OpenHAR datasets, respectively.

In a recent study, Hohman et al. [32] conducted by 30 experts from Apple specialized in efficient models, provides crucial insights into the practical challenges and

strategies for deploying models on resource-constrained devices. The study emphasizes starting with lightweight models before applying advanced compression techniques, as this often results in more reliable and efficient models. Compression experts highlight the importance of initial architecture optimization, noting that *"If you're trying to reduce the size, model compression is effective, but more like a last resort. If you have a model that is just overkill, [compression] can shave off 20-40%, but just changing architecture to something smaller will have a way bigger impact."*. Besides, the study also underscores the challenges of compression techniques. Methods such as quantization and pruning can sometimes fail, leading to wasted effort. As one practitioner remarked, *"[Compression] works super well sometimes, but sometimes totally fails. Each project is too dependent, recipes tend to only work in some situations."*. There is no *"silver bullet"* in model compression; the success of these techniques varies widely. Moreover, it is also denoted that there is a significant gap between theoretical efficiency gains reported in papers and practical performance on actual hardware. *"Don't expect what is "efficient" in a paper to exactly work in practice."* warned one expert. This highlights the necessity of extensive iterative testing and validation on the target hardware. The study concludes that a strategic and iterative approach, tailored to specific hardware and use cases, is essential for successfully deploying efficient models on edge devices.

A summary of the prominent studies using attention mechanisms or model compression techniques for sensor-based HAR is presented in Table 2.1 and 2.2. While many of these studies report higher recognition success, they often overlook the importance of resource consumption or do not achieve significant reductions in computational overhead. In contrast, our models achieve not the highest, but still high, recognition performance while significantly reducing memory usage and computational complexity for resource-constrained environments. For example, the Pamap2 model presented in [46] achieves f1-scores as high as 99%. However, this high recognition performance comes with millions of FLOPs, making it impractical for deployment on devices with limited resources. Our model, on the other hand, achieves only 5% less f1-score (94%), while requiring approximately 35x less FLOPs, demonstrating a much more efficient use of resources.

Table 2.1. Studies using attention mechanism and/or model compression for sensor-based HAR.

Ref.	Year	Attention mechanism	Lightweight model design	Knowledge distillation	Parameter quantization	Model pruning
[49]	2018	+				
[83]	2019	+				
[92]	2021	+				
[50]	2021	+				
[78]	2022	+				
[89]	2022	+				
[93]	2023	+				
[94]	2023	+				
[111]	2021		+			
[120]	2022		+			
[122]	2024		+			
[46]	2023			+		
[128]	2023			+		
[37]	2022				+	
[140]	2022				+	
[35]	2023				+	+
[141]	2023				+	
[39]	2023				+	
Our work	2024	+	+	+	+	+

Existing literature on sensor-based HAR highlights several limitations and open issues, including high computational demands and limited efficiency. Current models often enhance performance using attention without considering computational overhead, making them unsuitable for resource-constrained devices [50], [91–93]. Additionally, many studies lack generalizability as they often rely on a single dataset, limiting the robustness and adaptability of their models. Despite improvements, there is still room for optimization, as resource constraints continue to limit the feasibility of deploying deep models like DeepConvLSTM, which performs well but requires significant computational resources [92, 93]. Additionally, while attention mechanisms improve model performance, they often increase computational and memory demands, limiting their feasibility for real-time applications on resource-limited devices [49–51], [77, 78], [83]. Besides, compression techniques such as quantization and pruning, although useful in

reducing model size, sometimes fail to provide consistent results due to multiple factors, such as mismatches between hardware and deep architecture and model sensitivity to precision changes. Therefore, these techniques may not always achieve the theoretical efficiency gains expected [33], [37, 38], [140, 141], [144]. Even efficient models like SqueezeNet, known for their reduced resource consumption, still face challenges when deployed on devices with very limited memory, typically in the range of a few MBs.

In addressing these challenges in sensor-based HAR, this thesis integrates attention mechanisms and knowledge distillation to enhance the performance of lightweight deep learning models. We specifically utilized both DeepConvLSTM, known for its high performance, and SqueezeNet, recognized for its efficiency, to create and evaluate significantly smaller versions of these models. Unlike many studies that rely on simple CNNs or shallow architectures with just a few layers, we applied SqueezeNet, a well-known efficient CNN, to sensor-based HAR, demonstrating its potential in this domain. Additionally, we employed DeepConvLSTM, which is renowned for its strong performance in HAR tasks. To ensure generalizability, we evaluated our models across four diverse datasets: Opportunity, Sensors, Wisdm, and Pamap2, whereas many previous studies have been limited to a single dataset. This comprehensive approach demonstrated the robustness and adaptability of our methods. Furthermore, we compared our boosted models using two different model compression techniques, pruning and quantization, to evaluate performance and resource consumption comprehensively. In summary, this thesis stands out in the literature for its comprehensive approach, combining attention mechanisms and knowledge distillation to boost lightweight models into highly efficient ones. Furthermore, we compared the performance of our improved models with two different model compression techniques, pruning and quantization, to evaluate performance and resource consumption comprehensively. By integrating multiple techniques and testing across multiple datasets and two architectures, our work provides a comprehensive approach for deploying high-performing, resource-efficient HAR models on resource-limited devices, making it a significant contribution to the field of sensor-based human activity recognition.

Table 2.2. Performance results of studies using attention mechanism and/or model compression for sensor-based HAR. NA: Not available.

Ref.	Year	Recognition success	Resource usage
[49]	2018	Pamap2: 90% f1-score Daphnet Gait: 84% f1-score Skoda: 89% f1-score	NA
[83]	2019	HHAR: 96% f1-score Skoda: 93% f1-score Pamap2: 89% f1-score	NA
[92]	2021	Pamap2: 95% accuracy Daphnet Gait: 92% accuracy Skoda: 96% accuracy Sensors: 99% accuracy Daily: 92% accuracy	NA
[50]	2021	Wisdm: 99% accuracy Pamap2: 93% accuracy Opportunity: 83% accuracy Unimib-SHAR: 79% accuracy	Wisdm: 2.33M params Pamap2: 3.51M params Opportunity: 1.57M params Unimib-SHAR: 2.40M params
[78]	2022	UT dataset: 99% f1-score Pamap2: 97% f1-score Opportunity: 76% f1-score	UT dataset: 1.04M params and 9.04M FLOPs Pamap2: 1.22M params and 10.57M FLOPs Opportunity: 0.90M params and 7.20M FLOPs
[89]	2022	Pamap2: 92% f1-score Wisdm: 97% f1-score UCI-HAR: 97% f1-score Unimib-SHAR: 75% f1-score	Pamap2: 0.86M params Wisdm: 0.42M params UCI-HAR: 0.34M params Unimib-SHAR: 0.39M params
[93]	2023	Wisdm: 98% accuracy UCI-HAR: 96% accuracy	Wisdm: 1.55M params and 2.31 ms inference time UCI-HAR: 1.60M params and 3.05 ms inference time
[94]	2023	Pamap2: 96% f1-score	Pamap2: 0.304 s inference time
[111]	2021	UCI-HAR: 98% accuracy	NA
[120]	2022	Pamap2: 75% f1-score Daphnet Gait: 61% f1-score Opportunity: 39% f1-score Wisdm: 83% f1-score Skoda: 99% f1-score DSADS: 88% f1-score	Pamap2: 37.27K params and 19.13M FLOPs Daphnet Gait: 19.12K params and 2.08M FLOPs Opportunity: 165.19K params and 14.75M FLOPs Wisdm: 16.18K params and 1.47M FLOPs Skoda: 67.09K params and 19.82M FLOPs DSADS: 99.37K params and 55.68M FLOPs
[122]	2024	Pamap2: 98% f1-score DSADS: 94% f1-score RealDisp: 98% f1-score USC-HAD: 90% f1-score	Pamap2: 40K params, 20.18M MACs and 27.45 ms inference time DSADS: NA RealDisp: NA USC-HAD: NA
[46]	2023	Opportunity: 81% f1-score Pamap2: 99% f1-score	Opportunity: 0.308MB params and 39.5M FLOPs Pamap2: 0.308MB params and 45.51M FLOPs
[128]	2023	RealDisp: 93% accuracy	RealDisp: 2.55M params and 0.11G FLOPs
[37]	2022	Wisdm: 99% accuracy Unimib-SHAR: 86% accuracy UCI-HAPT: 86% accuracy	Wisdm: 6.21KB memory Unimib-SHAR: 23.16KB memory UCI-HAPT: 7.53KB memory
[140]	2022	Wisdm: 92% accuracy MobiAct: 98% accuracy	Wisdm: 392KB memory and 3100mWh energy / 32KB memory and 130mWh energy MobiAct: 56KB memory and 2000mWh energy / 12KB memory and 220mWh energy
[35]	2023	UCI-HAR: negligible loss of accuracy	UCI-HAR: energy saving between 31% and 37% for CNN networks and up to 45% for LSTM networks
[141]	2023	UCI-HAR: negligible loss of accuracy	UCI-HAR: energy savings of nearly 30%
[39]	2023	Wisdm: 98% accuracy MobiAct: 92% accuracy OpenHAR: 95% accuracy	Wisdm: 392KB memory and 3100mWh energy / 32KB memory and 130mWh energy MobiAct: 56KB memory and 2000mWh energy / 12KB memory and 220mWh energy OpenHAR: 71MB memory and 1530mWh energy / 1MB memory and 210mWh energy
Our work	2024	Opportunity: 71% f1-score Sensors: 96% f1-score Wisdm: 82% f1-score Pamap2: 94% f1-score	Opportunity: 9K params, 501K FLOPs and 37KB memory Sensors: 8K params, 1040K FLOPs and 97KB memory Wisdm: 2K params, 16K FLOPs and 11KB memory Pamap2: 14K params, 1303K FLOPs and 168KB memory

Additionally, several frameworks have been proposed to improve the efficiency of deep learning models on resource-constrained devices [145] including TensorFlow Lite [146], OpenEI [147], Core ML [148], Caffe2 [149], and MXNet [150]. In sensor-based HAR literature, we observe that TensorFlow Lite is the most popular tool [35], [39,40], [116], [140]. It improves the efficiency of deep models for deployment on IoT devices such as mobiles, microcontrollers, and other edge devices [33]. In addition to facilitating the conversion of the initial model into a lighter version with minimal accuracy loss, it also supports various model compression techniques, such as pruning and quantization. However, despite its popularity, TensorFlow Lite has certain downsides, detailed in Section 2.4.3.1.

2.4.3.1. TensorFlow Lite Issues, Limitations, and Constraints. TensorFlow Lite [33] is a specialized, lightweight version of TensorFlow designed to enable machine learning on mobile, embedded, and edge devices. While it addresses constraints such as latency, privacy, connectivity, size, and power consumption, TensorFlow Lite has inherent limitations and challenges impacting its usability and performance. These can be summarized as follows:

(i) Operator Compatibility and Custom Operators: TensorFlow Lite supports a limited set of TensorFlow operations, meaning not all TensorFlow models can be directly converted to TensorFlow Lite models [146]. This restriction necessitates modifications for models with unsupported operations, complicating the deployment process and limiting the use of standard runtime options [151]. When models require custom operators, TensorFlow Lite allows their inclusion, but this approach has significant downsides. Custom operators increase the model size due to additional dependencies outside the built-in TensorFlow Lite library, which is particularly problematic for resource-constrained devices. They can also degrade performance and necessitate modifications to the model runtime environment, further complicating the deployment process and reducing the ability to use standard runtime services [151]. While custom operators provide a way to include unsupported operations, they come with trade-offs in model size, performance,

and deployment complexity.

(ii) Lack of On-Device Training Support for Microcontrollers: TensorFlow Lite supports on-device training for mobile devices, enabling real-time model updates and personalization [152]. However, this capability is unavailable for microcontrollers, limiting their ability to adapt and improve models based on new data collected on the device [153]. For example, in certain sensor-based HAR systems, real-time adaptation can be crucial. If a user changes their activity patterns or introduces new activities, on-device training would enable the model to quickly learn and recognize these new patterns. Without this capability on microcontrollers, the model cannot update based on the latest data, leading to decreased accuracy over time. Given the global deployment of more than 250 billion microcontrollers as of November 2023 [14], this limitation is significant, affecting a vast number of potential applications. Additionally, TensorFlow Lite supports only a limited set of microcontrollers, even for the inference phase [153]. For instance, the processor of the HummingBoard Pro does not support TensorFlow Lite, whereas the HummingBoard Ripple is compatible with TensorFlow Lite [154].

(iii) Tool and Library Maturity: Although TensorFlow Lite integrates well within the TensorFlow ecosystem, the tools and libraries available for TensorFlow Lite are not as extensive or mature as those for TensorFlow itself. This limitation affects the ease with which developers can build, test, and deploy models using TensorFlow Lite, particularly for complex applications requiring robust support and extensive tooling.

Despite these limitations, TensorFlow Lite continues to evolve rapidly. Developers and researchers must consider these constraints and explore alternative approaches or supplementary methods to optimize model performance within the limitations of TensorFlow Lite. It is important to note that such optimizations applied using TensorFlow Lite can only reduce the model size to a certain point, typically up to 10 times smaller. This may still not be small enough for extremely resource-constrained devices.

3. DEEP LEARNING MODELS AND DATASETS

This chapter outlines our methodology, starting with an overview of the baseline models: Deep Convolutional and LSTM Recurrent Neural Network and SqueezeNet. Then, the datasets used throughout the thesis are presented: Opportunity, Sensors, Wisdm, and Pamap2. We describe the datasets' diversity and the challenges they present. Next, we detail the implementation and training protocols, including optimization algorithms and learning rates. Finally, we discuss our model evaluation process, focusing on performance metrics like accuracy, precision, recall, and f1-score, as well as resource consumption metrics such as FLOPs, model sizes, and inference time. These metrics are essential for assessing the models' feasibility for deployment on resource-constrained devices.

3.1. Baseline Models

In this section, we describe two architectures in detail: DeepConvLSTM, a hybrid model designed explicitly for sensor-based HAR, and SqueezeNet, which is proposed for minimal resource consumption. We provide a thorough overview of their structures, features, and the reasons for their selection in our study.

3.1.1. Deep Convolutional and LSTM Recurrent Neural Network

The Deep Convolutional and LSTM Recurrent Neural Network architecture [41] has gained widespread recognition for its effectiveness in sensor-based HAR [91], [155–157]. This model exploits the advantages of CNNs for feature extraction and LSTM networks for capturing temporal dependencies.

The DeepConvLSTM model processes 2D-time window sensor data inputs, where the temporal dimension is represented by height (H) and the sensor signals by width (W). For example, if the window size is set to 1-second and 3-dimensional accelerometer

and magnetometer sensors are used with a sampling rate of 50 Hz, the input width is 6 (representing the x, y, and z axes of both sensors), and the height is 50 (corresponding to the 50 samples in 1 second). This input is fed to DeepConvLSTM, more specifically into four consecutive convolutional layers with 64 filters each, followed by two consecutive LSTM layers, each with 128 hidden units. Finally, a softmax function is applied to the output of the last LSTM layer to generate class probabilities for the input data.

We selected the DeepConvLSTM architecture for its robust performance in distinguishing between closely related activities, such as *open drawer and close drawer* [18]. Its ability to handle wearable sensor data from multiple sources makes it ideal for sensor-based HAR tasks. The convolutional layers effectively extract spatial features, while the LSTM layers capture temporal dependencies, making them suitable for recognizing complex activity patterns over time. One of the primary challenges of deploying DeepConvLSTM models on resource-constrained devices, such as mobile and wearable sensors, is their high computational and memory demands. To address this, various methods can be employed to reduce the complexity of the original DeepConvLSTM model and create a more lightweight version, including structural simplification by reducing the number of layers or decreasing the channel dimensions in each layer.

3.1.2. SqueezeNet

SqueezeNet [42] is a highly efficient CNN model designed to achieve AlexNet-level accuracy with 50x fewer parameters. Its compact size and low computational requirements make it ideal for resource-constrained applications, such as mobile devices and embedded systems. The architecture employs a novel *fire module* design, which reduces the number of parameters while maintaining high accuracy. This module consists of a squeeze layer with 1x1 filters and an expand layer with both 1x1 and 3x3 filters. SqueezeNet processes standard 2D image inputs, with the height and width representing the spatial dimensions and the depth corresponding to colour channels (e.g., RGB channels). The model begins with an initial convolution layer with a relatively large kernel size, followed by max-pooling. A series of *fire modules* and additional

max-pooling layers are then employed, concluding with a convolutional layer, global average pooling, and a softmax function to generate class probabilities. Note that in this thesis, we evaluated SqueezeNet using only the Opportunity and Sensors datasets. The reason is that the number of positions and, consequently, the input width of the Wisdm and Pamap2 datasets are insufficient to feed SqueezeNet. Therefore, for the Wisdm and Pamap2 datasets, we present results only for DeepConvLSTM.

We selected SqueezeNet for its low resource consumption to test on a pure CNN architecture, enabling us to compare results across different baseline models. Among the manually designed efficient CNN architectures, including MobileNets [96] and ShuffleNet [97], SqueezeNet stands out as the least resource-intensive [98]. This makes it particularly suitable for applications with strict resource constraints, where minimizing computational load and memory usage is crucial. Its efficiency makes it highly effective for various tasks, including real-time image classification, where computational resources are limited. However, to the best of our knowledge, SqueezeNet has not yet been explored in the sensor-based HAR domain.

In addition to the original SqueezeNet, two enhanced versions have been developed: SqueezeNet with simple bypass and SqueezeNet with complex bypass, as presented in [42]. The simple bypass version adds shortcut connections around specific *fire modules*, helping these modules learn the difference between their input and output, similar to ResNet [139]. This approach changes how the model's parameters are regularized and can improve both accuracy and training efficiency. However, these bypass connections can only be added to certain *fire modules* where the number of input and output channels match. For cases where the channels do not match, a complex bypass is used, which includes a 1x1 convolution layer to adjust the number of output channels, adding more parameters to the model. Given our aim to maximize performance while keeping resource use low, we chose to implement SqueezeNet with a simple bypass in our experiments. In our implementation, bypass connections are added around *fire modules* 3, 5, 7, and 9, helping to balance parameter efficiency and performance gains.

3.2. Datasets

In this thesis, we use four sensor-based HAR datasets: Opportunity [86], Sensors [65], Pamap2 [80], and Wisdm [85]. Datasets differ in terms of the number of activity sets, the number of samples, and the signal properties. The distribution of activities within each dataset varies significantly. The Sensors dataset has a perfectly balanced class distribution, whereas Wisdm has an extreme imbalance, and Opportunity and Pamap2 have imbalanced distributions. Although these imbalances in the datasets complicate the study, they also enhance its generalizability, enabling the assessment of model performance across diverse activity distributions. Table 3.1 provides a comprehensive summary of the datasets. Besides, to have a clear understanding of the placement of sensors used in this study, we present the locations of the sensors for each dataset in Figure 3.1. Then, to illustrate the complexity and separability of the activities within these datasets, we provide two-dimensional t-SNE visualizations (Figure 3.2). These t-SNE visualizations highlight the distribution and clustering of different activities, showcasing the inherent challenges in accurately recognizing activities.

Table 3.1. Details about datasets.

Dataset	**# of activities**	**Class/activity names**	**Sampling rate (Hz)**	**Sensor names**	**# of positions**	**Class distribution**
Opportunity [86]	17	open door 1, open door 2, close door 1, close door 2, open fridge, close fridge, open dishwasher, close dishwasher, open drawer 1, close drawer 1, open drawer 2, close drawer 2, open drawer 3, close drawer 3, clean table, drink from cup, toggle switch	30	accelerometer, gyroscope, magnetometer	5	mean: 6%, min: 4%, max: 21%
Sensors [65]	7	walking, standing, jogging, sitting, biking, upstairs, downstairs	50	accelerometer, gyroscope, magnetometer	5	mean: 14%, min: 14%, max: 14%
Wisdm [85]	6	jogging, walking, upstairs, downstairs, sitting, standing	20	accelerometer	1	mean: 17%, min: 1%, max: 40%
Pamap2 [80]	12	rope jumping, lying, sitting, standing, walking, running, cycling, nordic walking, ascending stairs, descending stairs, vacuum cleaning, ironing	100	accelerometer, gyroscope, magnetometer	3	mean: 8%, min: 2%, max: 12%

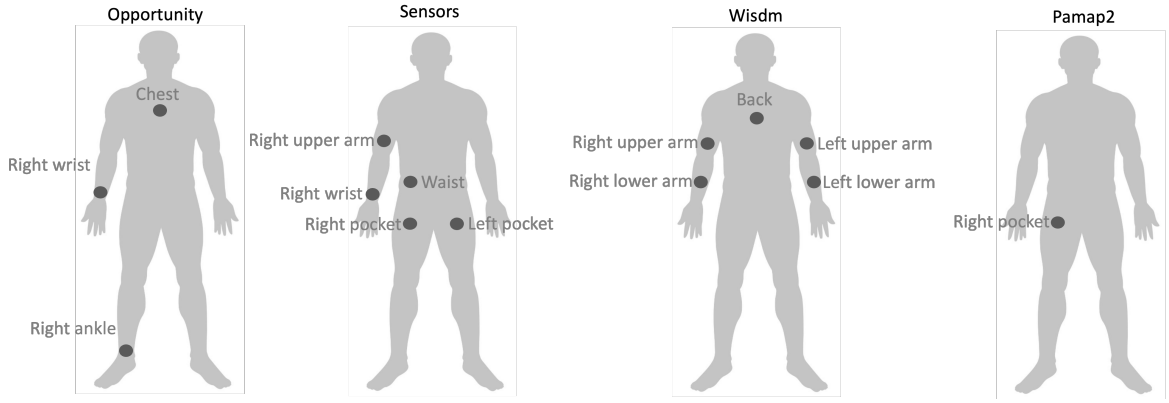


Figure 3.1. Positions of wearable sensors.

3.2.1. Opportunity

The Opportunity dataset [86] comprises real-world daily life activities data gathered from a variety of sensors, including wearable sensors, object sensors, and ambient sensors. Sensor-based HAR studies primarily focus on data collected through body-worn sensors to categorize 4 locomotion activities and 18 micro activities. The data is collected from 4 participants, and during data collection, 5 IMUs are placed on the upper body: left lower arm, left upper arm, right upper arm, right lower arm, and the back of the torso, while the remaining two sensors are placed on the users' shoes.

Although the dataset encompasses 18 micro activities, there is a dominant null class constituting over 70% of the data, resulting in a highly imbalanced dataset. In this study, we focused on 17 micro activities by excluding the null class, a practice also observed in the dataset's original study [86] and some other research works [158, 159]. However, even with the null class extracted, the dataset remains challenging due to the presence of very similar movements, such as opening and closing the door. Additionally, in this study, we used 5 IMUs on the upper body, as they possess comparable devices with similar embedded sensors (3-dimensional accelerometer, gyroscope, and magnetometer). The sampling rate of the experiments remained the same as the dataset's initial setting at 30 Hz.

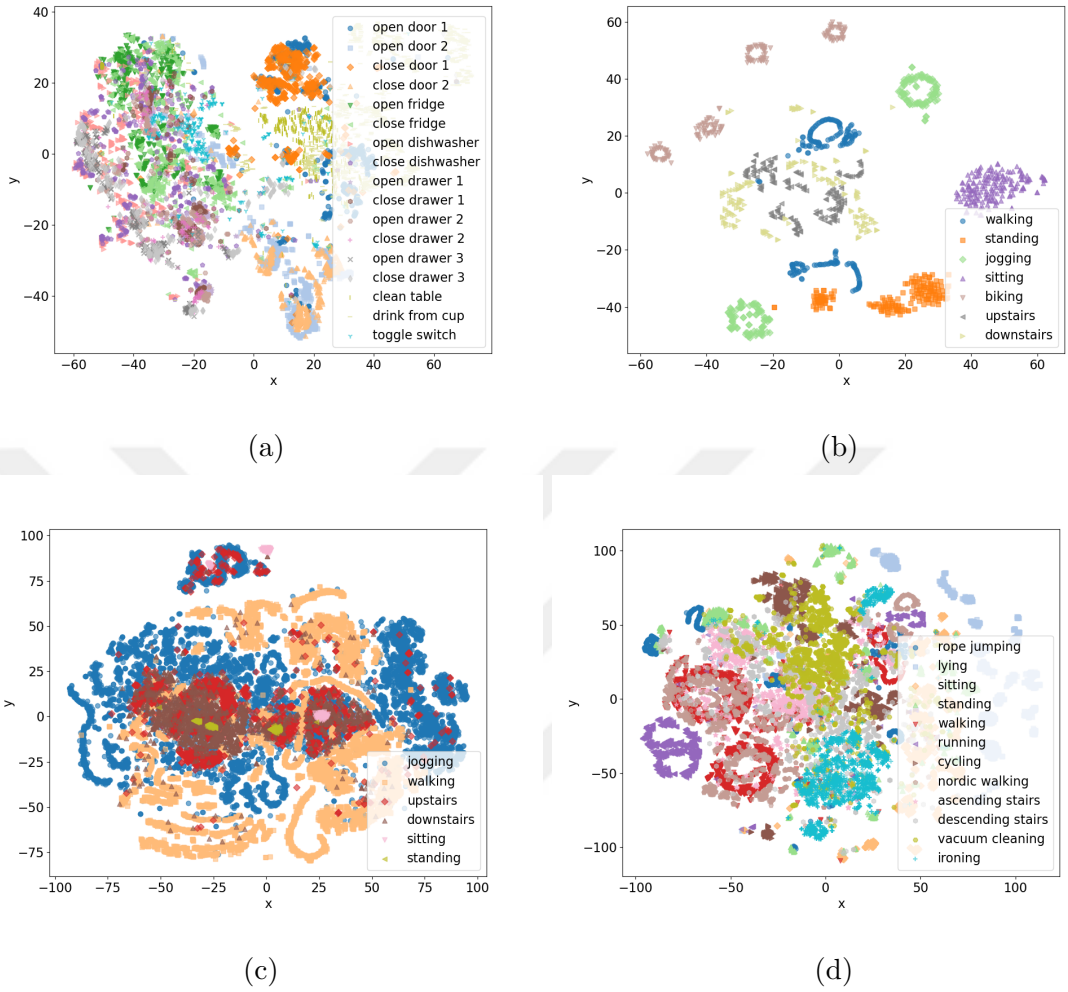


Figure 3.2. Two-dimensional t-SNE visualizations of HAR datasets. (a) Opportunity, (b) Sensors, (c) Wisdm, (d) Pamap2.

3.2.2. Sensors

Sensors dataset [65] contains 7 physical activities collected from 10 participants using 5 IMUs (smartphones) placed in the different body positions, which are in the right and left jeans pocket, on the belt position towards the right leg, on the right upper arm and the right wrist. The sampling rate used while collecting data is 50 Hz. In this study, we investigate all activities utilizing the complete set of sensor data collected at the initial sampling rate of 50 Hz.

3.2.3. Wisdm

The Wireless Sensor Data Mining Dataset (Wisdm) [85] is a popular benchmark dataset for studies on sensor-based HAR. Data was collected from a 3-axis accelerometer embedded in a smartphone positioned in the user's pocket while they performed six daily activities. During the data collection process, a frequency of 20 Hz is used. Even though the activities are simple, this dataset is significantly imbalanced compared to the other three datasets. For instance, an activity class constitutes approximately 40% of the data, but another activity class constitutes only 1% of the data.

3.2.4. Pamap2

In the Physical Activity Monitoring Dataset (Pamap2) [80], sensor data are collected from nine participants for 18 different daily activities. In this dataset, sensor signals from both simple activities, such as sitting and walking, and more complex activities, such as ironing and jumping rope, are collected from various body parts. During data collection, participants are asked to wear three IMUs on their dominant wrist, chest, and ankle, along with a heart rate monitor. Each IMU is equipped with a 3-axis accelerometer, gyroscope, and magnetometer motion sensors. The sampling rate of IMUs is 100 Hz, and that of the heart rate monitor operates at 9 Hz.

In this study, we investigated 12 activities common to all participants (with the remaining 6 being optional activities, such as driving a car and playing soccer). We utilized 27 signals for each time step from IMUs positioned over 3 body regions, emphasizing the capture of participants' movements. These signals consisted of 9 readings for each body part: accelerometer (x, y, and z), gyroscope (x, y, and z), and magnetometer (x, y, and z), all collected at 100 Hz sampling rate.

3.3. Implementation Details and Evaluation Metrics

In this thesis, we investigate the baseline DeepConvLSTM architecture using the Opportunity, Sensors, Wisdm, and Pamap2 datasets. However, SqueezeNet is only investigated with the Opportunity and Sensors datasets. The number of body positions and sensors, and thus the input width, of the Wisdm and Pamap2 are insufficient for SqueezeNet. Specifically, in the DeepConvLSTM architecture, while the height dimension decreases through the layers, the width dimension remains constant. For example, the input width of Pamap2 is 27, and it stays 27 throughout all convolutional layers in this architecture, so having a smaller width is not an issue. On the other hand, in SqueezeNet, due to the presence of max-pooling layers in the architecture, both the height and width dimensions of the input decrease through the layers. The Wisdm dataset, with an input width of 3 (one sensor on one body part with x, y, z axes), and the Pamap2 dataset, with an input width of 27 (three sensors on three body parts with x, y, z axes), are insufficient as input for SqueezeNet. However, the Sensors and Opportunity datasets, both with an input width of 45, can be used as input for SqueezeNet. Therefore, while results using DeepConvLSTM are presented for all four datasets, only the results for Sensors and Opportunity are presented using SqueezeNet.

In our experiments for four datasets, we utilize 1-second time windows for data segmentation and apply a split ratio of 60% for training, 20% for testing, and 20% for validation. For both SqueezeNet and DeepConvLSTM architectures, the Adam optimizer is used with its default settings, along with a batch size of 64 and a learning rate of 0.001. For the training, we set a high epoch number (200 epochs) and we also incorporate early stopping by tracking validation accuracy, with a patience of 10 epochs. Then, we restore the model with the best weights. The models are implemented using TensorFlow, TensorFlow Lite and Keras (version 2.12.0). The experiments are conducted on a computer with the following specifications: Apple M3 Pro chip with 11-core CPU and 14-core GPU, 18 GB of RAM, and running macOS Sonoma 14.1.

In this thesis, we prioritize enhancing recognition performance while also considering the resource constraints of target devices. To this end, in Chapters 4, 5, and 6, we investigate the total number of trainable parameters (params) and floating-point operations (FLOPs), which significantly impact computational resource utilization, energy consumption, and memory usage. These factors are crucial for edge devices, as they directly affect battery life during both training and testing. Unlike metrics such as inference time, which can vary from device to device, params and FLOPs provide device-independent measures. This makes them more reliable for comparing resource consumption across different deep models. Consequently, these metrics are frequently used in the literature to assess whether a model is advantageous or disadvantageous regarding resource utilization.

However, in Chapter 7, we focused on quantization and pruning methods and used TensorFlow Lite models instead of regular TensorFlow models. In quantization, the number of parameters does not change, but their precision does, which means that params and FLOPs may not accurately reflect resource consumption across models. In pruning, some of the model's parameters are set to zero. However, this does not mean that a model pruned by 50% will have 50% fewer parameters, as the positions of the zeroed-out values must also be stored. The more scattered these zero values are, the more space the model occupies in memory, affecting the model's overall size. Therefore, in Chapter 7, we evaluated the models based on memory footprint and inference time. For each model, we present the inference time for 1000 samples. Inference time impacts energy usage and thus battery life, while model size indicates how much memory the model occupies, which is critical for devices with limited memory. These metrics provide a clearer picture of the model's resource utilization in real-world scenarios, especially for edge devices.

We use several metrics to evaluate recognition performance. To reflect more accurate model performance on imbalanced datasets, we use the macro f1-score (denoted as f1-score), defined as the arithmetic mean of the f1-scores of all classes. Additionally, we provide the accuracy metric for a general overview of the model's performance.

Furthermore, we present precision and recall to offer a detailed understanding of the recognition performance.

4. INTEGRATING CONVOLUTIONAL BLOCK ATTENTION MODULES IN DEEP LEARNING FOR SENSOR-BASED HAR

This chapter explores the integration of the Convolutional Block Attention Module (CBAM) into deep learning models for sensor-based HAR. The primary goal is to enhance model performance while considering the resource constraints of wearable IoT devices and wearable technology. CBAM is designed to highlight the most informative features in the data, potentially improving both accuracy and efficiency. This section evaluates the benefits of applying CBAM by examining the effects of channel and spatial attention mechanisms across different datasets and deep learning architectures. The objective is to determine whether these mechanisms can boost recognition accuracy and resource efficiency without significantly increasing computational demands.

4.1. Convolutional Block Attention Module

Woo et al. [52] proposed the CBAM, an attention mechanism that boosts performance by enhancing informative channels and important regions of intermediate features. The primary research examines the effects of CBAM by analyzing its performance using commonly utilized datasets in computer vision, such as ImageNet [138] and Cifar-10 [137]. However, they did not utilize sensor data in these experiments. In addition to its potential to enhance performance, it represents a promising opportunity for sensor-based activity recognition models incorporating a convolutional layer. This is due to the notably minimal parameter demand of attention modules and their straightforward integration on any convolutional layer. CBAM comprises two sub-modules: channel attention (CA) module and spatial attention (SA) module (Figure 4.1). As the CBAM name suggests, the modules are designed to be applied after convolutional layers.

4.1.1. Channel Attention Module

The Channel Attention (CA) module aims to weigh essential channels (C) of features with greater importance than unimportant ones in convolutional layers. It uses max-pooling (for a sharper effect) and average-pooling (for a smoothing effect) in the spatial dimension to process the input features. These are followed by a Multi-layer Perceptron (MLP) mapped by a reduction ratio (r) and then a sigmoid activation function ($\sigma(.)$). The reduction ratio is the key parameter that controls the degree of dimensionality reduction to create a trade-off between computational efficiency and attention accuracy through a shared MLP module in the channel attention mechanism. A smaller reduction ratio can increase the expressive capacity of the channel attention mechanism but with the cost of higher computational complexity and visa-versa for a larger reduction ratio. Depending on the specific application, the reduction ratio should be tuned to achieve the best balance between attention performance and computational efficiency.

To be more precise, given an input feature $F \in \mathbb{R}^{C\times H\times W}$ to compute the channel attention map $M_c(F) \in \mathbb{R}^{C\times 1\times 1}$, we first compute two vectors via max-pooling and average-pooling in the spatial dimension ($avgPool^s(F)$ and $maxPool^s(F) \in \mathbb{R}^{C\times 1\times 1}$). Then, these vectors are passed one by one as input to the shared MLP, which has C neurons in the input layer, C/r neurons in the hidden layer, and C neurons in the output layer. After getting two output vectors from MLP, the average-pooling mask and max-pooling mask are obtained. These masks are added together using element-wise summation, denoted as $\oplus$. Then, to obtain the final CA map $M_c(F)$, a sigmoid activation layer is applied to map values within the range of 0 and 1. In summary, the CA map can be computed as

$$M_c(F) = \sigma(MLP(avgPool^s(F)) \oplus MLP(maxPool^s(F))). \tag{4.1}$$

Finally, to compute CA based new input feature ($F_c \in \mathbb{R}^{C\times H\times W}$), we perform element-wise multiplication, denoted as $\otimes$, where each channel element in F is multiplied by

its corresponding channel attention value, expressed as

$$F_c = M_c(F) \otimes F. \tag{4.2}$$

4.1.2. Spatial Attention Module

The Spatial Attention (SA) module aims to improve convolutional layer features by highlighting relevant regions and suppressing irrelevant ones. It consists of three consecutive operations. First, two tensors are computed using max-pooling and average-pooling across channels of input feature F ($avgPool^c(F)$ and $maxPool^c(F)$ $\in \mathbb{R}^{1\times H\times W}$). Second, two tensors are concatenated and passed as an input to the convolution layer ($Conv(.)$) with a kernel size of $k \times k$ to generate one channel feature map. Third, the sigmoid activation is applied to the output in order to obtain the final SA map $M_s(F)$. In overall, the SA map can be computed as

$$M_s(F) = \sigma(Conv(f^{kxk}[avgPool^c(F); maxPool^c(F)])). \tag{4.3}$$

To obtain the SA based new input feature ($F_s \in \mathbb{R}^{C\times H\times W}$), F is multiplied element-wise by the spatial attention map computed. This computation can be expressed as

$$F_s = M_s(F) \otimes F. \tag{4.4}$$

Finally, given an input feature F, the overall CBAM flow is expressed as

$$F_{cs} = M_s(M_c(F) \otimes F) \otimes (M_c(F) \otimes F), \tag{4.5}$$

where we first apply CA to the input feature F and then apply SA to the CA applied F feature.

CBAM uses both spatial and cross-channel relationships of features by successively combining channel and spatial attention. To be more precise, it highlights helpful channels and strengthens local regions that are more informative. The CBAM has a lightweight design. CA module requires $2 * C * (C/r) + C + (C/r)$ parameters to learn in shared MLP, and SA module requires $k * k * 2$ parameters with k as the kernel size of the convolutional layer. From this point of view, it is understandable that the improvements made possible by CBAM are not related to the model's increased capacity

but to efficient feature refining. It should be noted that r for CA and k for SA module are the only parameters that can be selected experimentally based on the problem.

There are multiple options to use CA and/or SA modules, such as using both in parallel, using one of them, using both but SA first, and using both with CA first. Based on the experimental findings reported in the main paper of the presented CBAM method, we decided to investigate CA alone, SA alone, and CA-SA (CSA (first CA and then SA)) as presented in Figure 4.1. As suggested in the main paper, the attention modules can be applied either after one of the convolutional layers or after every convolutional layer. In this study, we conduct experiments regarding how to use this attention mechanism from the perspective of resource-constrained devices while using sensor data since CBAM has negligible parameter numbers [52]. Therefore, we designed our experiments to apply the attention mechanism to only a single layer at a time.

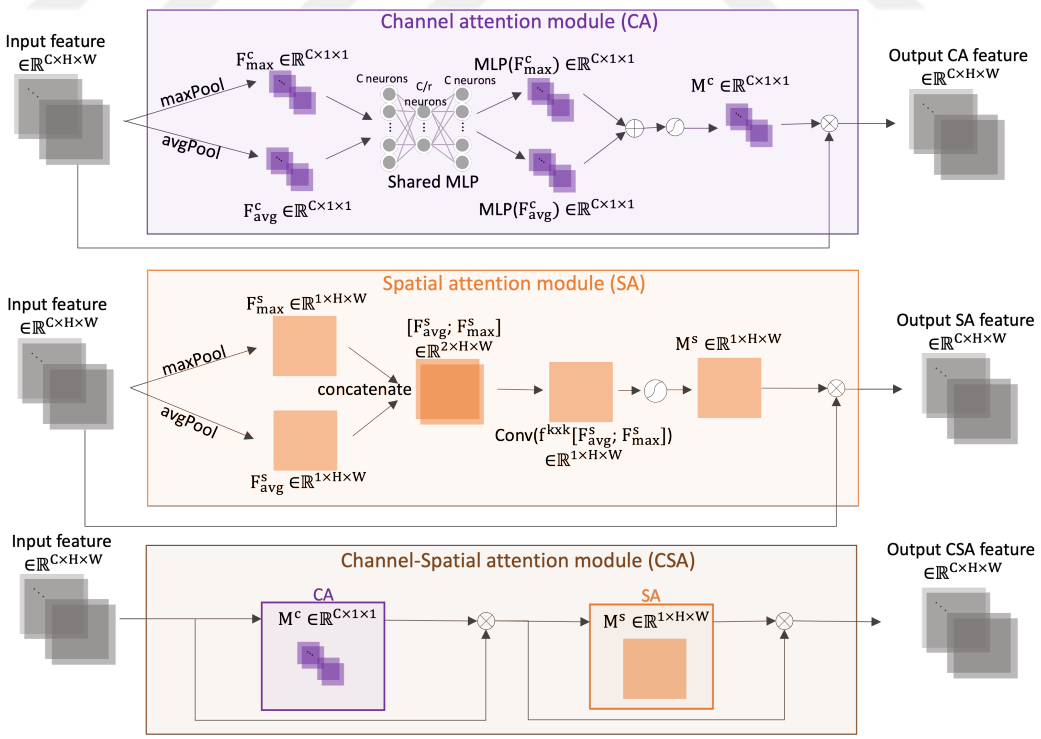


Figure 4.1. Application of channel attention and spatial attention mechanisms individually and in combination using CBAM.

To apply CBAM, we utilize the well-established DeepConvLSTM model, known for its capability to model temporal dependencies in HAR studies and the SqueezeNet model, known for its efficiency in resource-constrained environments. We integrate the CBAM into the models to enhance their performance by concentrating on significant channels and spatial regions within the convolutional layers. By addressing the temporal aspect of sensor-based HAR and improving the convolutional layers through channel and spatial attention, we aim to develop more accurate and robust models for sensor-based HAR.

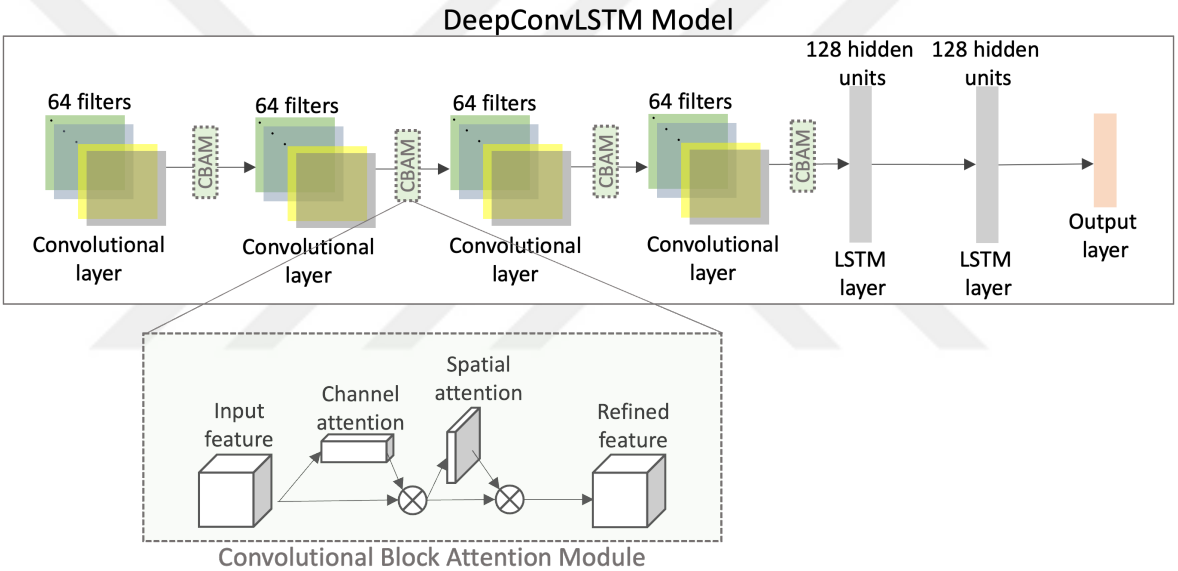


Figure 4.2. DeepConvLSTM architecture with the CBAM.

The architectures incorporating DeepConvLSTM and SqueezeNet with CBAM are illustrated in Figures 4.2 and 4.3, respectively. In the DeepConvLSTM architecture, the CA, SA, and CSA modules are applied after the first, second, third, or fourth convolutional layers (one layer at a time). Each CA module operates on 64 channels per convolutional layer, re-scaling the significance of each channel's features, which are extracted along the temporal axis. The SA module pools across channels, creating a single map for each signal-temporal feature pair. Further details on how averaging along different dimensions affect time series properties can be found in [160].

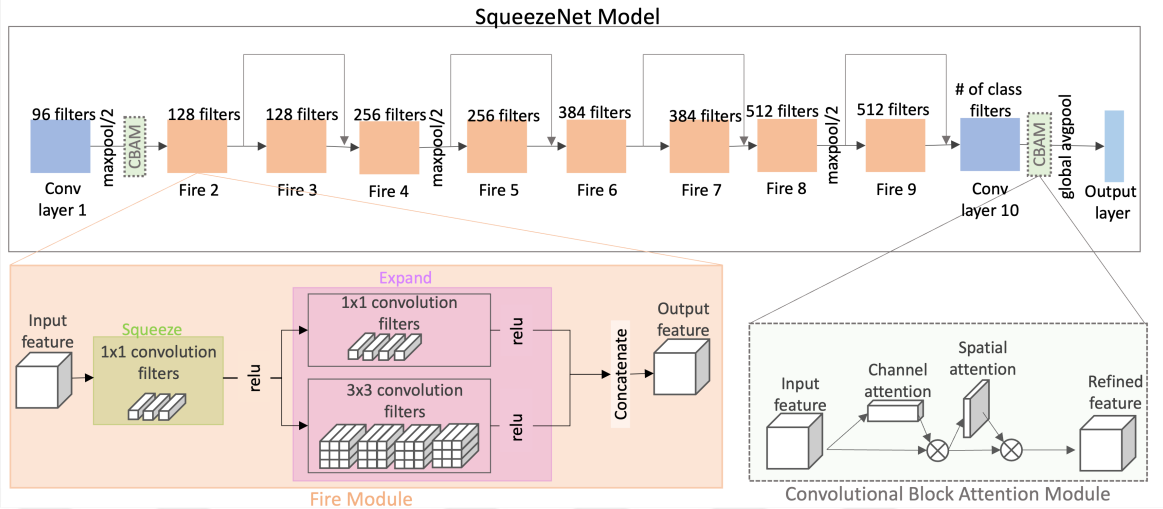


Figure 4.3. SqueezeNet architecture with the CBAM.

For the SqueezeNet architecture, the CA, SA, and CSA modules are applied to the convolutional layers right before or after the fire modules. The first CA module processes 96 channels, while the second CA module handles the number of classes in the recognition task. Similar to DeepConvLSTM, each channel contains features extracted along the temporal axis. The SA module in SqueezeNet also utilizes pooling across channels to generate a single map for each signal-temporal feature pair.

4.2. Experimental Results and Analysis of Attention Modules

In this section, we present the experimental results for activity recognition performance using the DeepConvLSTM network and SqueezeNet with the CBAM method. We begin by evaluating channel and spatial attention separately to address RQ1 and then in combination to adress RQ3. Attention is applied to various convolutional layers, using different kernel sizes for spatial attention and different reduction ratios for channel attention to further examining RQ2. Table 4.1 provides descriptions of the abbreviations used for different attention settings.

While aiming to enhance recognition performance with attention mechanisms, it is crucial to avoid increasing the resource consumption of models expected to be

deployed on resource-constrained edge devices, such as mobile phones and wearables for HAR. The number of parameters in the model directly impacts the computational resources, such as CPU cycles, required during both training and testing phases, which in turn affects the battery consumption of the target devices. Thus, as the number of parameters in the model increases, so does resource consumption and computation time. Therefore, when evaluating each attention mechanism, it is important to consider the additional parameters it introduces to the model.

Table 4.1. Abbreviations used for attention settings in the experiments.

Abbreviation	**Description**
CAx(r)	Channel attention module applied to x^{th} convolutional layer with a reduction ratio r
SAx(k)	Spatial attention module applied to x^{th} convolutional layer with a kernel size k
CSAx(r)(k)	First channel and then spatial attention module applied to x^{th} convolutional layer with a reduction ratio r and kernel size k for channel and spatial attention, respectively

Table 4.2. Additional params for DeepConvLSTM in attention experiment. The number of parameters of the DeepConvLSTM model is 816,337 for Opportunity, 1,414,087 for Sensors, 393,030 for Wisdm, and 3,053,644 for Pamap2.

	Reduction ratio (r)	**Kernel size (k)**	**# of additional parameters**
CAx(r)	2		4,192
	4		2,128
SAx(k)		3	18
		7	98
CSAx(r)(k)	2	3	4,210
		7	2,146
	4	3	4,290
		7	2,226

Before assessing the recognition performance of the models, we present the number of additional parameters required by attention in Table 4.2 and Table 4.3 for DeepConvLSTM and SqueezeNet, respectively. In the DeepConvLSTM architecture, the number of channels for all layers is fixed, and it is 64. Consequently, the number

of extra parameters required by the CA module remains the same across all datasets. Additionally, since the number of variables in the SA module depends on the kernel size k parameter, it does not vary according to the dataset or the output size of the layer to which it is applied (which will be the input to the attention module). Therefore, considering DeepConvLSTM, the number of additional parameters is consistent for all four datasets. In contrast, the number of channels in SqueezeNet varies from layer to layer. While the number of channels in layer 1 remains constant at 96, the number of channels in layer 2 varies according to the number of classes in the recognition task. Thus, the number of additional parameters required by the CA module does not vary for layer 1 across datasets, but it does vary for layer 2. The number of additional parameters required by the SA module depends solely on the kernel size k, so it does not change according to the layer or the dataset.

Table 4.3. Additional params for SqueezeNet in attention experiment. The parameter number of the SqueezeNet model is 734,737 for Opportunity and 729,607 for Sensors.

				# of additional parameters	
	Reduction ratio (r)	**Kernel size (k)**	**Attention layer**	**Opportunity**	**Sensors**
CAx(r)	2		layer 1	9,360	9,360
			layer 2	297	52
	4		layer 1	4,728	4,728
			layer 2	157	22
SAx(k)		3	layer 1 and 2	18	18
		7	layer 1 and 2	98	98
CSAx(r)(k)	2	3	layer 1	9,378	9,378
			layer 2	315	70
		7	layer 1	4,746	4,746
			layer 2	175	40
	4	3	layer 1	9,458	9,458
			layer 2	395	150
		7	layer 1	4,826	4,826
			layer 2	255	120

For example, when applying the CSA module with a reduction ratio of 2 and a kernel size of 3 to layer 1 in the SqueezeNet architecture for the Opportunity dataset,

the total number of parameters required by the CSA module is 9,378. This is the sum of the 9,360 parameters required by the CA module ($2 \times C \times (C/r) + C + (C/r) = 2 \times 96 \times (96/2) + 96 + (96/2) = 9,360$) and the 18 parameters required by the SA module ($k \times k \times 2 = 3 \times 3 \times 2 = 18$)).

Our experiments show that the CBAM attention mechanism slightly increases the total number of parameters in both the DeepConvLSTM and SqueezeNet models. However, the number of parameters required for spatial attention is significantly lower compared to channel attention. Therefore, if channel and spatial attention mechanisms achieve similar recognition performances, it is more advantageous to select the spatial attention mechanism in terms of resource consumption.

4.2.1. Evaluating the Impact of Channel Attention

In this section, we examine the impact of channel attention by using two reduction ratios (r: 2 and 4) at different depths (after layer 1, layer 2, layer 3, and layer 4). Abbreviations used for reduction ratios r and the number of layers after which attention is applied x are listed in Table 4.1. As outlined in Section 4.1.1, channel attention identifies the most significant channels while filtering out less relevant or confusing ones in a given input feature. Additionally, the channel attention parameter r determines the complexity of the MLP in the module and the number of additional parameters required.

According to Table 4.2, for DeepConvLSTM, CA requires 4,192 additional parameters with $r = 2$ and 2,128 additional parameters with $r = 4$, regardless of the layer and dataset. Considering SqueezeNet (Table 4.3), depending on the dataset, channel attention with $r = 2$ adds 9,360 parameters to layer 1 and up to 297 parameters to layer 2, while channel attention with $r = 4$ adds 4,728 parameters to layer 1 and up to 157 parameters to layer 2. When comparing two channel attention experiments with similar performance results, it is preferable to choose the one with a higher r value and consequently fewer additional parameters.

The performance results obtained using CA for the Opportunity, Sensors, Pamap2, and Wisdm datasets using DeepConvLSTM and for the Opportunity and Sensors datasets using SqueezeNet are presented in Figure 4.4. The dark red bar represents the base model (DeepConvLSTM or SqueezeNet) without attention, and the green bar represents the best performance obtained in terms of validation accuracy using the CA module. Note that, in all experiments, SqueezeNet is only used with the Opportunity and Sensors due to the insufficient input width of the Wisdm and Pamap2 for this architecture, as detailed in Section 3.3.

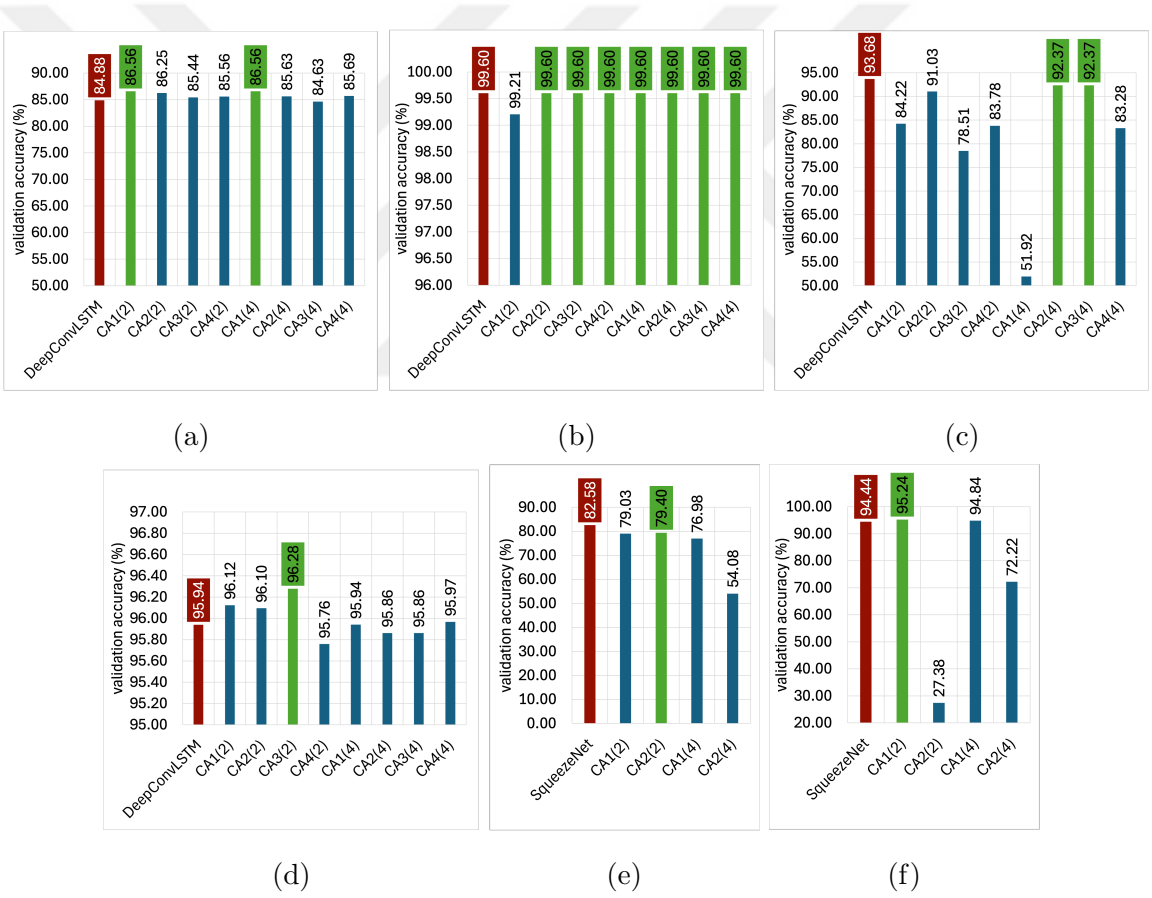


Figure 4.4. Impact of channel attention on recognition performance. (a) Opportunity-DeepConvLSTM, (b) Sensors-DeepConvLSTM, (c) Wisdm-DeepConvLSTM, (d) Pamap2-DeepConvLSTM, (e) Opportunity-SqueezeNet, (f) Sensors-SqueezeNet.

For the Opportunity dataset using DeepConvLSTM, seven out of eight CA experiments outperform the 84.88% baseline achieved without attention. The highest

improvement is 1.68%, achieved with CA1(2) and CA1(4). The lowest result is with CA3(4), indicating a slight decrease. For the Wisdm dataset using DeepConvLSTM, the performance without attention is 93.68%, and with channel attention, it ranges between 51.92% and 92.37%. The best result of CA is lower than that obtained without attention. For the Pamap2 dataset using DeepConvLSTM, the performance without attention is 95.94%, and with CA, it ranges between 95.76% and 96.28%. The highest performance is observed with CA3(2), demonstrating a small but notable improvement. The lowest performance is with CA1(4), showing a marginal decrease. In the Sensors dataset using DeepConvLSTM, the recognition performance of the model without attention is 99.60%. All but one of the CA experiments match this high performance, and given the already very high performance of this dataset, no performance increase is observed.

For the Opportunity dataset using SqueezeNet, unlike in DeepConvLSTM, CA does not provide any improvement over the baseline. However, in the Sensors dataset, a very slight improvement (0.80%) is observed. Additionally, when looking at the base performance results, DeepConvLSTM outperforms SqueezeNet for both the Opportunity and Sensors datasets.

Applying CA does not provide any performance improvement for three out of six dataset-architecture pairs. However, for the remaining three pairs, the improvements range between 0.34% and 1.68%. These improvements come with only a minimal increase in parameter count, demonstrating the efficiency and applicability of CA in resource-constrained environments. For example, in the Opportunity dataset using DeepConvLSTM, the base performance of 84.88% increases to 86.56% by applying CA1(4), requiring only 2,128 additional parameters over the 816,337 baseline DeepConvLSTM.

4.2.2. Evaluating the Impact of Spatial Attention

In this section, we investigate how spatial attention affects recognition performance. The SA module is applied at different layers using various kernel sizes. The abbreviations for the kernel sizes k and the layers where attention is applied are provided in Table 4.1.

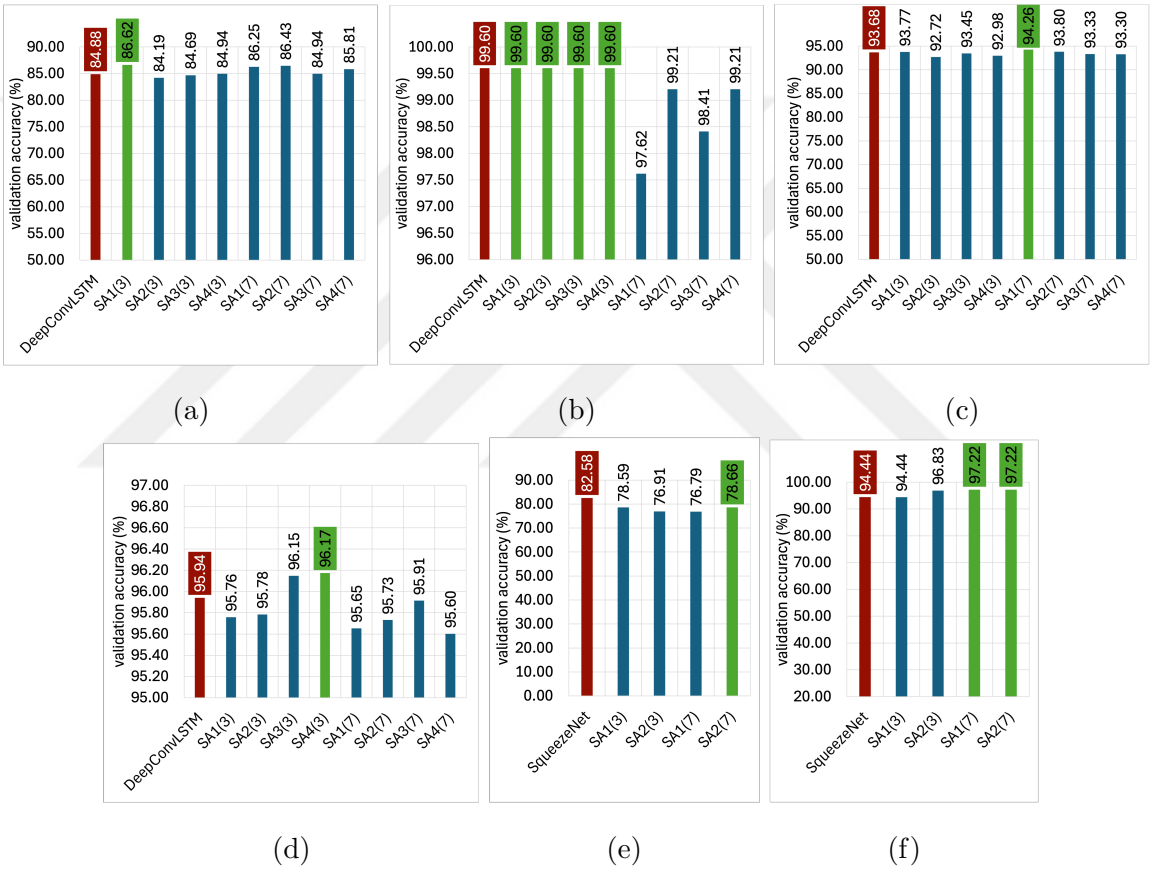


Figure 4.5. Impact of spatial attention on recognition performance.
(a) Opportunity-DeepConvLSTM, (b) Sensors-DeepConvLSTM,
(c) Wisdm-DeepConvLSTM, (d) Pamap2-DeepConvLSTM,
(e) Opportunity-SqueezeNet, (f) Sensors-SqueezeNet.

Figure 4.5 presents the impact of SA on the recognition performance for the Opportunity, Sensors, Pamap2, and Wisdm datasets with DeepConvLSTM and for the Opportunity and Sensors datasets with SqueezeNet. The dark red bar represents the base model's performance without attention, while the green bar indicates the optimal

performance achieved using the SA module.

For the Opportunity dataset using DeepConvLSTM, all SA experiments, except for SA2(3), outperform the 84.88% baseline achieved without attention. The highest improvement is 1.73%, achieved with SA1(3). For the Wisdm dataset, the performance without attention is 93.68%, and with spatial attention, it ranges between 90.45% and 95.27%. The highest result of SA is higher than that obtained without attention. For the Pamap2 dataset, with SA, a slight increase is obtained. However, the performance of most of the SA experiments is lower than the baseline DeepConvLSTM performance. In the Sensors dataset, the recognition performance without attention is 99.60%. All SA experiments match or fall short of this performance. In DeepConvLSTM experiments, while using CA does not improve performance for Wisdm and Sensors, using SA resulted in no improvement only for the Sensors dataset.

For the Opportunity dataset using SqueezeNet, SA does not improve the performance of the baseline model. However, for the Sensors dataset, the best results obtained using SA (SA1(7) and SA2(7)) showed a significant improvement (2.78%).

Overall, SA improves performance across most datasets and models. Compared to CA, in almost all cases where the attention-based performance exceeds the baseline, the improvement achieved with SA is greater than that obtained with CA. Additionally, considering that the number of parameters required for spatial attention is significantly lower compared to channel attention, SA can be considered more advantageous.

4.2.3. Evaluating the Impact of Channel-Spatial Attention

In this section, we evaluate the effect of the combined channel and spatial attention (CSA) module on recognition performance. Results obtained are presented in Figure 4.6.



Figure 4.6. Impact of channel-spatial attention on recognition performance. (a) Opportunity-DeepConvLSTM, (b) Sensors-DeepConvLSTM, (c) Wisdm-DeepConvLSTM, (d) Pamap2-DeepConvLSTM, (e) Opportunity-SqueezeNet, (f) Sensors-SqueezeNet.

For the Opportunity using DeepConvLSTM, the highest improvement is 1.99%, achieved with CSA2(4)(3). For the Wisdm, performance without attention is 93.68%, and with CSA, it achieves a slight increase of 0.23%. In the Pamap2, CSA results in an increase from 95.94% to 96.64%. Finally, for the Sensors, the best performance obtained using CSA is 100%. Considering SqueezeNet, CSA does not increase baseline performance for Opportunity but achieves a significant increase (2.78%) for Sensors.

When comparing CSA with individual channel and spatial attention, CSA generally provides higher improvements. For instance, in the Opportunity dataset, the highest improvement with channel attention alone was 1.68%, while spatial attention alone yielded 1.73%. The combined approach with CSA achieves a better result (improvement of 1.99%), indicating the enhanced effect of combining both attention mechanisms. Furthermore, CSA achieves these improvements with a manageable increase in parameter count, slightly higher than the channel attention's additional parameter count, making it an efficient choice for resource-constrained environments. In conclusion, CSA consistently boosts performance across different datasets and models, providing an effective way to enhance recognition accuracy while efficiently managing resource consumption.

Table 4.4. Performance of base models and best attention models.

Architecture	Dataset	Attention	Validation accuracy(%)	Validation f1-score(%)	Test accuracy(%)	Test f1-score(%)	Params	FLOPs
DeepConvLSTM	Opportunity	No	84.88	79.46	82.95	77.86	816,337	100,512,000
		Yes: CSA2(4)(3)	86.87	82.05	84.38	80.11	818,483 (0.26%↑)	100,809,432 (0.30%↑)
	Sensors	No	99.60	99.63	99.21	99.07	1,414,087	211,910,400
		Yes: CSA3(2)(7)	100.00	100.00	98.41	98.15	1,418,377 (0.30%↑)	212,699,896 (0.37%↑)
	Wisdm	No	93.68	91.49	93.91	91.04	393,030	2,987,520
		Yes: SA1(7)	94.26	92.11	93.48	90.40	393,128 (0.02%↑)	3,003,072 (0.52%↑)
	Pamap2	No	95.94	95.49	96.75	96.49	3,053,644	294,243,840
		Yes: CSA3(4)(7)	96.64	96.35	97.03	96.93	3,055,870 (0.07%↑)	295,326,144 (0.37%↑)
SqueezeNet	Opportunity	No	82.58	77.97	81.39	77.35	734,737	28,504,644
		Yes: CA2(2)	79.40	70.12	76.04	66.43	735,034 (0.04%↑)	28,505,850 (0.00%↑)
	Sensors	No	94.44	94.47	91.27	90.23	729,607	48,256,088
		Yes: SA1(7)	97.22	97.32	95.24	94.65	729,705 (0.01%↑)	48,479,188 (0.46%↑)

Table 4.4 provides a comprehensive comparison of the performance results between base models and their best-performing attention-improved models for six dataset-architecture pairs. If multiple experiments achieve the same performance for a dataset-architecture pair, the one with the least number of parameters is selected. We observe that the attention mechanism, especially the combination of channel and spatial attention (CSA), results in improved recognition success. For instance, the DeepConvLSTM model with CSA2(4)(3) on the Opportunity dataset shows an increase in the validation f1-score from 79.46% to 82.05%, and a test f1-score improvement from 77.86% to 80.11%, with only a 0.26% increase in parameter count and 0.30% in FLOPs. A comparable improvement is observed with the SqueezeNet model on the Sensors dataset.

Applying SA1(7) increases the test f1-score from 90.23% to 94.65%, while the parameters increase minimally by 0.01% and the FLOPs rise by less than 0.5%. On the other hand, the DeepConvLSTM model on the Pamap2 dataset with CSA3(4)(7) shows only a slight increase in validation f1-score from 95.49% to 96.35% and in test f1-score from 96.49% to 96.93%, with a 0.07% increase in parameters.

However, for the DeepConvLSTM models on the Sensors and Wisdm datasets, although there are small increases in validation accuracy, the test performance does not improve over the base model. In fact, the test f1-score for the DeepConvLSTM on the Sensors dataset with CSA3(2)(7) decreases from 99.07% to 98.15%, and on the Wisdm dataset with SA1(7), it decreases from 91.04% to 90.40%. Furthermore, the application of CA2(2) to the SqueezeNet model on the Opportunity dataset leads to a substantial performance drop, with the validation f1-score decreasing from 77.97% to 70.12% and the test f1-score from 77.35% to 66.43%. In conclusion, while attention mechanisms can enhance model performance, they need to be applied judiciously, as their effectiveness can vary significantly across different datasets and architectures.

4.3. Conclusion and Insights

In this chapter, we investigated the impact of channel and spatial attention modules, both individually with varying parameters like reduction ratios and kernel sizes, and in combination to better understand their effects. We evaluated the integration of these modules into the hybrid sensor-based HAR architecture DeepConvLSTM and the resource-efficient SqueezeNet architectures across four datasets: Opportunity, Sensors, Pamap2, and Wisdm. Our findings demonstrate that applying CBAM attention mechanisms, specifically CA, SA, and CSA, can improve the recognition performance with a negligible (less than 0.5%) parameter and FLOPs overhead.

For instance, the DeepConvLSTM model enhanced with CSA2(4)(3) on the Opportunity dataset showed a remarkable increase in validation f1-score from 79.46% to 82.05%, and a test f1-score from 77.86% to 80.11%, with only a 0.26% increase in

parameter count. Such improvements indicate that the CBAM effectively emphasizes the most informative features while reducing the influence of less relevant ones. Additionally, the individual use of CA and SA modules also provided notable performance gains. However, the effectiveness of these attention mechanisms varies across different datasets and architectures. While some configurations led to performance improvements, others did not yield significant gains or even caused performance drops. For example, the SqueezeNet model on the Opportunity dataset experienced a decrease in validation f1-score from 77.97% to 70.12% with the application of CA2(2), highlighting the sensitivity of attention mechanisms to specific model and dataset characteristics. This underscores the necessity of careful parameter tuning of attention mechanisms adapted to the characteristics of the dataset and model architecture.

The chapter also examined the resource implications of integrating CBAM into the models in detail. Although attention mechanisms introduce additional parameters, their number is relatively minimal, making them suitable for resource-constrained environments. Considering best-performing models, the additional number of parameters required by attention ranges between 0.01% and 0.30% of the model's total number of parameters. This balance between performance enhancement and resource efficiency makes attention mechanisms a valuable addition to sensor-based HAR models.

5. ACHIEVING RESOURCE-EFFICIENT HAR WITH LIGHTWEIGHT DEEP MODELS ENHANCED BY ATTENTION

This chapter investigates the impact of attention mechanisms on the performance and resource efficiency of the models at different sizes. With the increasing use of mobile and wearable devices, there is a growing need to deploy efficient deep-learning models that can function under strict resource constraints. In the previous chapter, we examined the effect of attention mechanisms on large models. While attention mechanisms improved performance, the parameter count of these models remains too high for deployment, even in the inference phase, particularly on wearable IoT devices. Therefore, this chapter shifts the focus on exploring the impact of attention modules on smaller models that are more feasible, requiring less computation and memory, for wearable devices.

We aim to determine whether integrating attention mechanisms into lightweight models can enable them to achieve recognition performance comparable to or surpass that of larger models while maintaining lower resource consumption. Additionally, we assess a moderate-sized model to analyze how the impact of attention mechanisms varies with model size. Smaller models, which lack the inherent complexity of larger models, may benefit more significantly from attention mechanisms, potentially enhancing their feature extraction capabilities and overall performance.

5.1. Designing Resource-Efficient Models Using Base Models

The DeepConvLSTM model and SqueezeNet models are difficult to deploy on resource-limited devices, such as wearables, due to their high parameter counts and computational demands. In order to overcome this challenge, we designed three versions of DeepConvLSTM: the lightweight model (LM), the moderate-size model (MM), and the original model (OM). These variants are designed to explore resource con-

sumption and performance trade-offs when using CBAM (explained in Section 4.1) and knowledge distillation (KD) (explained in Section 6.1). To create the *LM* and *MM* models without relying on the pre-trained *OM*, there are different methods employed, such as reducing model depth by eliminating layers, decreasing model width by lowering the number of CNN filters and LSTM hidden units, using 1D convolutions instead of 2D, and applying pooling. In our study, we chose to reduce the model's width (not the depth) to preserve the integrity of the DeepConvLSTM architecture and ensure the consistent application of the same attention module across all three versions. As a result, we reduced the number of filters in the convolutional layers from 64 to 8 for *MM* and 4 for *LM*, and the number of hidden units in the LSTM from 128 to 16 for *MM* and 8 for *LM*. These design choices are detailed in Table 5.1, showing the structural configurations of the different DeepConvLSTM versions. This approach allows us to investigate the balance between model simplification and performance when applying CBAM attention and KD techniques.

Table 5.1. Structure of the DeepConvLSTM models at different sizes.

	Lightweight Model (*LM*)	**Moderate Model (*MM*)**	**Original Model (*OM*)**
Input	$(1 \times H \times W)$	$(1 \times H \times W)$	$(1 \times H \times W)$
Conv 1	[5×1 kernel size, 4 filters]	[5×1 kernel size, 8 filters]	[5×1 kernel size, 64 filters]
Conv 2	[5×1 kernel size, 4 filters]	[5×1 kernel size, 8 filters]	[5×1 kernel size, 64 filters]
Conv 3	[5×1 kernel size, 4 filters]	[5×1 kernel size, 8 filters]	[5×1 kernel size, 64 filters]
Conv 4	[5×1 kernel size, 4 filters]	[5×1 kernel size, 8 filters]	[5×1 kernel size, 64 filters]
LSTM 1	[8 units]	[16 units]	[128 units]
LSTM 2	[8 units]	[16 units]	[128 units]
Dense (Output)	(# of classes)	(# of classes)	(# of classes)

Similar to DeepConvLSTM, the initial number of filters in the layers of SqueezeNet is set to values ranging between 96 and 16. These values are then adjusted by a constant known as the *compression rate*. In the original SqueezeNet, this compression rate is set to 1, referred to as the original model (*OM*) in our study. To create the *MM* and *LM* versions of the SqueezeNet architecture, we set the *compression rates* to 1/8 and 1/16, respectively. These *compression rates* ensure that the ratio of the

number of parameters in *LM* and *MM* relative to the number of parameters in *OM* for DeepConvLSTM is maintained in SqueezeNet as well. The structures of these three versions are detailed in Table 5.2. Note that *fire module* denoted in the table is already explained in Section 3.1.2.

Table 5.2. Structure of the SqueezeNet models at different sizes.

	Lightweight Model (*LM*)	**Moderate Model (*MM*)**	**Original Model (*OM*)**
Input	$(1 \times H \times W)$	$(1 \times H \times W)$	$(1 \times H \times W)$
Conv 1	[3×3 kernel size, 6 filters]	[3×3 kernel size, 12 filters]	[3×3 kernel size, 96 filters]
MaxPool 1	[2×2, stride 2]	[2×2, stride 2]	[2×2, stride 2]
Fire 2	[1×1 squeeze, 1 filter; 1×1 expand, 4 filters; 3×3 expand, 4 filters]	[1×1 squeeze, 2 filters; 1×1 expand, 8 filters; 3×3 expand, 8 filters]	[1×1 squeeze, 16 filters; 1×1 expand, 64 filters; 3×3 expand, 64 filters]
Fire 3	same as Fire 2	same as Fire 2	same as Fire 2
Fire 4	[1×1 squeeze, 2 filters; 1×1 expand, 8 filters; 3×3 expand, 8 filters]	[1×1 squeeze, 4 filters; 1×1 expand, 16 filters; 3×3 expand, 16 filters]	[1×1 squeeze, 32 filters; 1×1 expand, 128 filters; 3×3 expand, 128 filters]
MaxPool 4	[2×2, stride 2]	[2×2, stride 2]	[2×2, stride 2]
Fire 5	[1×1 squeeze, 2 filters; 1×1 expand, 8 filters; 3×3 expand, 8 filters]	[1×1 squeeze, 4 filters; 1×1 expand, 16 filters; 3×3 expand, 16 filters]	[1×1 squeeze, 32 filters; 1×1 expand, 128 filters; 3×3 expand, 128 filters]
Fire 6	[1×1 squeeze, 3 filters; 1×1 expand, 12 filters; 3×3 expand, 12 filters]	[1×1 squeeze, 6 filters; 1×1 expand, 24 filters; 3×3 expand, 24 filters]	[1×1 squeeze, 48 filters; 1×1 expand, 192 filters; 3×3 expand, 192 filters]
Fire 7	same as Fire 6	same as Fire 6	same as Fire 6
Fire 8	[1×1 squeeze, 4 filters; 1×1 expand, 16 filters; 3×3 expand, 16 filters]	[1×1 squeeze, 8 filters; 1×1 expand, 32 filters; 3×3 expand, 32 filters]	[1×1 squeeze, 64 filters; 1×1 expand, 256 filters; 3×3 expand, 256 filters]
MaxPool 8	[2×2, stride 2]	[2×2, stride 2]	[2×2, stride 2]
Fire 9	same as Fire 8	same as Fire 8	same as Fire 8
Conv 10	[1×1, # of classes]	[1×1, # of classes]	[1×1, # of classes]
AvgPool 10 (Output)	(# of classes)	(# of classes)	(# of classes)

5.2. Impact of Attention Modules on Different Model Sizes

We performed a series of experiments to evaluate how attention modules influence different model sizes, focusing on the *LM*, *MM*, and *OM* variants of both DeepCon-

vLSTM and SqueezeNet architectures, addressing RQ4. To determine the impact of attention, we applied CA with reduction ratios of 2 and 4, and SA with kernel sizes of 3 and 7, as well as a combination of both CA and SA. For simplicity and clarity, we report results for the best-performing attention setting, referred to as *LM-Att*, *MM-Att*, and *OM-Att* throughout this thesis.

Figure 5.1 presents the impact of attention mechanisms on validation accuracy across different model sizes. The attention mechanisms significantly boosted the performance of the *LM*. For instance, in the Opportunity dataset, the accuracy for DeepConvLSTM improved from 71.38% (*LM*) to 77.29% (*LM-Att*), showing a 5.91% increase. Similarly, in the Sensors dataset, SqueezeNet's accuracy increased from 82.54% (*LM*) to 90.48% (*LM-Att*), an impressive 7.94% enhancement. This pattern of improvement is consistent across other datasets, with notable increases in Wisdm and Pamap2.

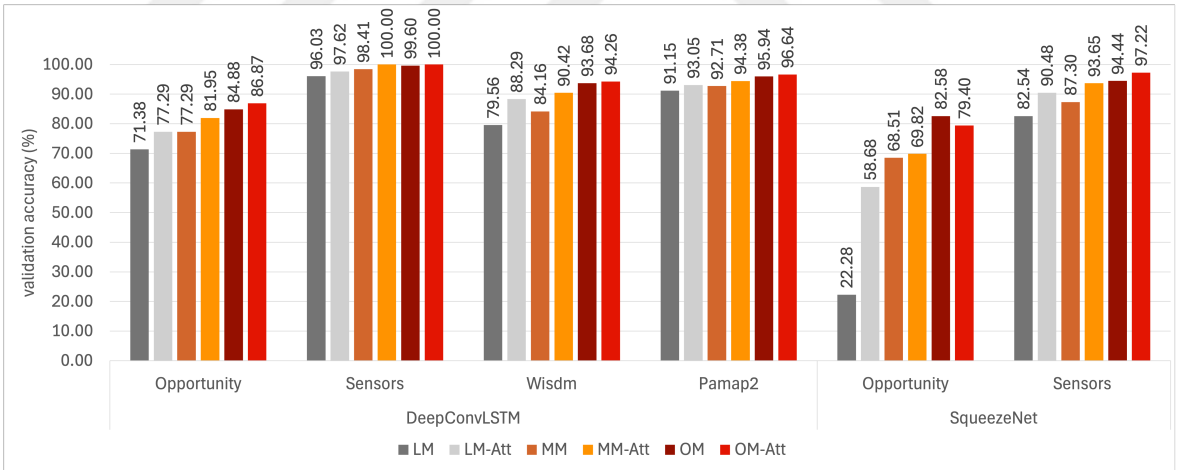


Figure 5.1. Performance results using base models and attention models.

When comparing these improvements for *MM*, the performance of *LM-Att* often approaches or exceeds that of the base *MM*. For example, in the Opportunity dataset, *LM-Att* achieves 77.29%, which is equal to *MM*'s 77.29%. In the Sensors dataset, *LM-Att* reaches 90.48%, surpassing *MM*'s 87.30% by a significant margin. This demonstrates that *LM* with attention mechanisms can not only match but also exceed the performance of *MM*, offering a more resource-efficient solution. Besides, we

observe that the attention mechanisms have a more noticeable impact on *LM* compared to *MM* and *OM*. For instance, in the Pamap2 dataset, the accuracy improvement from *MM* to *MM-Att* is 1.67% (from 92.71% to 94.38%), whereas the improvement for *LM* is more substantial. Besides, we observe that adding attention modules can enhance the performance of lightweight models, though the degree of improvement varies across datasets. This trend suggests that as the model size increases, the relative impact of attention mechanisms decreases. For lightweight models like *LM*, attention mechanisms appear to improve recognition performance noticeably. However, in larger models with better feature extraction capabilities, such as *OM*, attention mechanisms do not seem to significantly boost performance. Therefore, attention modules may be particularly effective in enhancing the feature extraction capabilities of lightweight models, potentially making them more efficient and capable of approaching or exceeding the performance of larger models.

Table 5.3. Comparison of datasets using LM with attention and LM, MM, and OM without attention.

Architecture	Dataset	Model Name	Test accuracy (%)	Test precision (%)	Test recall (%)	Test f1-score (%)	Params (K)	FLOPs (K)
DeepConvLSTM	Opportunity	**LM-Att**	**74.36**	**68.66**	**67.60**	**67.06**	**9.05**	**457.96**
		LM	68.95	62.54	61.26	61.04	9.04	450.00
		MM	76.29	70.28	69.87	69.81	23.66	1,677.60
		OM	82.95	78.22	78.03	77.86	816.34	100,512.00
	Sensors	**LM-Att**	**96.83**	**96.58**	**96.39**	**96.43**	**8.03**	**1,040.12**
		LM	95.63	95.39	95.21	95.13	7.99	932.40
		MM	95.63	95.26	95.40	95.28	26.69	3,506.40
		OM	99.21	99.07	99.09	99.07	1,414.09	211,910.40
	Wisdm	**LM-Att**	**86.87**	**81.29**	**79.88**	**80.36**	**1.80**	**16.46**
		LM	78.92	45.06	53.65	47.05	1.77	13.92
		MM	83.61	74.66	66.75	66.36	6.57	50.88
		OM	93.91	92.09	90.10	91.04	393.03	2,987.52
	Pamap2	**LM-Att**	**93.88**	**93.55**	**93.61**	**93.57**	**14.48**	**1,302.95**
		LM	91.33	91.73	89.63	90.51	14.46	1,283.04
		MM	92.64	92.30	91.62	91.89	52.44	4,847.04
		OM	96.75	96.63	96.38	96.49	3,053.64	294,243.84
SqueezeNet	Opportunity	**LM-Att**	**56.94**	**46.90**	**47.30**	**43.25**	**3.95**	**387.98**
		LM	21.59	1.27	5.88	2.09	3.85	316.22
		MM	66.21	59.52	57.18	57.21	13.29	827.76
		OM	81.39	78.02	77.75	77.35	734.74	28,504.64
	Sensors	**LM-Att**	**83.73**	**83.41**	**82.70**	**81.96**	**3.62**	**529.54**
		LM	79.76	78.11	78.39	77.59	3.52	528.70
		MM	85.71	85.16	84.93	83.72	12.64	1,388.99
		OM	91.27	90.22	90.40	90.23	729.61	48,256.09

The Table 5.3 compares the performance of lightweight models with attention (*LM-Att*) to models without attention (*LM*, *MM* and *OM*) for four datasets across various performance metrics. Notably, *OM* consistently outperforms *LM*, demonstrating the significant performance gap. For instance, Opportunity using DeepConvLSTM, *OM* achieves an test f1-score of 77.86%, significantly higher than *LM*'s 61.04%, showcasing a performance increase of 16.82%. Similarly, Sensors using SqueezeNet, *OM* reaches a test f1-score of 90.23%, while *LM* lags at 77.59%.

While *MM* also improves upon *LM*, it comes with a steep increase in resource consumption. For example, in the Pamap2 using DeepConvLSTM, *LM* requires 1,283.04K FLOPs and 14.46K trainable parameters. In contrast, *MM* needs 4,847.04K FLOPs and 52.44K parameters, approximately 3.8x more FLOPs and 3.6x more parameters than *LM*. This clearly indicates that higher performance from *MM* demands significantly more computational resources.

Applying attention mechanisms to *LM* (*LM-Att*) yields substantial performance gains with minimal additional resource consumption. For instance, in the Sensors dataset, *LM-Att* achieves a test accuracy of 96.83% and an f1-score of 96.43%, surpassing *MM*'s 95.63% accuracy and 95.28% f1-score with only a slight increase in FLOPs (from 932.40K to 1,040.12K) and parameters (from 7.99K to 8.03K). Similarly, in the Pamap2 dataset, *LM-Att* shows a test accuracy of 93.88% and an f1-score of 93.57%, closely matching *MM*'s 92.64% accuracy and 91.89% f1-score, but with far less computational demand.

In summary, the results clearly show that while more complex models (OM, *MM*) achieve higher performance, they require significantly more resources. However, *LM-Att* presents a compelling alternative, substantially enhancing performance over *LM* with only a minimal increase in resource consumption. This demonstrates the efficacy of attention mechanisms, suggesting that *LM-Att* models are an efficient choice for resource-constrained environments, as they balance high-performance metrics with lower computational costs.



Figure 5.2. Activity-based performance of LM, LM-Att and MM. (a) Opportunity-DeepConvLSTM, (b) Sensors-DeepConvLSTM, (c) Wisdm-DeepConvLSTM, (d) Pamap2-DeepConvLSTM, (e) Opportunity-SqueezeNet, (f) Sensors-SqueezeNet.

5.3. Activity-Based Impact

This section analyzes the impact of *MM* and *LM-Att* over *LM*. The results for each dataset-architecture pair are represented in Figure 5.2.

For the Opportunity-DeepConvLSTM, *LM-Att* shows significant improvements over the baseline *LM* in most of the activities. Although *LM-Att*'s overall f1-score performance approaches that of *MM*, it does not entirely surpass it. Specifically, for activities such as *opening the fridge*, *closing the fridge*, *closing the dishwasher*, and *closing drawer 1*, *LM-Att* performs better than *MM* by approximately 1% to 4%. However, *MM* significantly outperforms *LM-Att* in *opening drawer 1* and *closing drawer 2* by 16.09% and 12.18%, respectively. These two activities also exhibit low performance with the *LM* (47.19% and 45.98%).

For the Sensors-DeepConvLSTM, using *LM-Att* instead of *LM* results in the most considerable performance improvement for the *walking activity*, which initially had the lowest performance in *LM* (85.71%). Activities with a perfect performance score (100%) show no change, while the *biking activity*, initially at 98.70% with *LM*, reaches 100% with *LM-Att*. When comparing *MM* to *LM-Att*, it is evident that *LM-Att*'s performance is never lower than *MM* for any *activity*.

The figure obtained for the Wisdm-DeepConvLSTM pair reveals that *LM-Att* consistently outperforms *MM* across all activities. The activities *downstairs* and *standing*, which were not recognized by *LM*, see substantial performance improvements with *MM* (23.74% and 60.47%, respectively), and even further with *LM-Att*, reaching over 85.25% for *standing*. For the *sitting activity*, *LM*'s performance increases from 65.96% to 98.41% with *LM-Att*, a value that is 20.91% higher than that achieved with *MM*.

The Pamap2-DeepConvLSTM results shows significant performance enhancements with *LM-Att*, particularly for the *rope jumping activity*, which increases from 86.42% with *LM* to 96.93%, 8.53% higher than that achieved with *MM*. Moreover,

LM-Att improves performance by more than 2% in eight out of twelve activities (up to 10.21%), while *MM* only shows similar improvement in four activities (up to 4.96%).

The SqueezeNet architecture, which lacks an LSTM-like mechanism to capture temporal connections, performs poorly with the *LM* for the Opportunity dataset. However, incorporating attention in *LM* (*LM-Att*) results in significant performance improvements for seven activities, with f1-scores exceeding 50%. Despite this, these values remain noticeably lower than those achieved with *MM*. For instance, *LM-Att* outperforms *MM* only for the *close door 1 activity* by 5.64%. For other activities, *MM* consistently outperforms *LM-Att*, and in one case (*open dishwasher*), performance increases from 0% with *LM* to 48% with *MM*.

For the Sensors-SqueezeNet, very high performances are achieved for *sitting* and *standing* (100% and 98.63% respectively), with no changes observed with *LM-Att* or *MM*. For four activities, *MM* achieves higher performance improvements than *LM-Att*, ranging from 4.02% to 10.52%. However, the overall performance of *LM-Att* is higher than *MM* (83.72% vs. 81.96%), primarily because *LM-Att* significantly improves the lowest performing activity (*downstairs*) from 29.51% with *LM* to 62.96%, 18.96% higher than that with *MM*.

In Figure 5.3 we present confusion matrices before and after the use of attention in *LM* to examine the activities that are confused with each other in more detail. Although the number of samples is generally preferred, since the datasets are imbalanced, the colour of the dominant class would be almost completely dark. In contrast, the others would be almost white. Therefore, we colour the matrices using row-based normalization, with each cell showing the percentage of the activity in that row predicted as the activity in the corresponding column. Additionally, since the performance of Opportunity-SqueezeNet with *LM* was nearly %0 (see Table 5.3), we do not include its comparison with *LM-Att*.

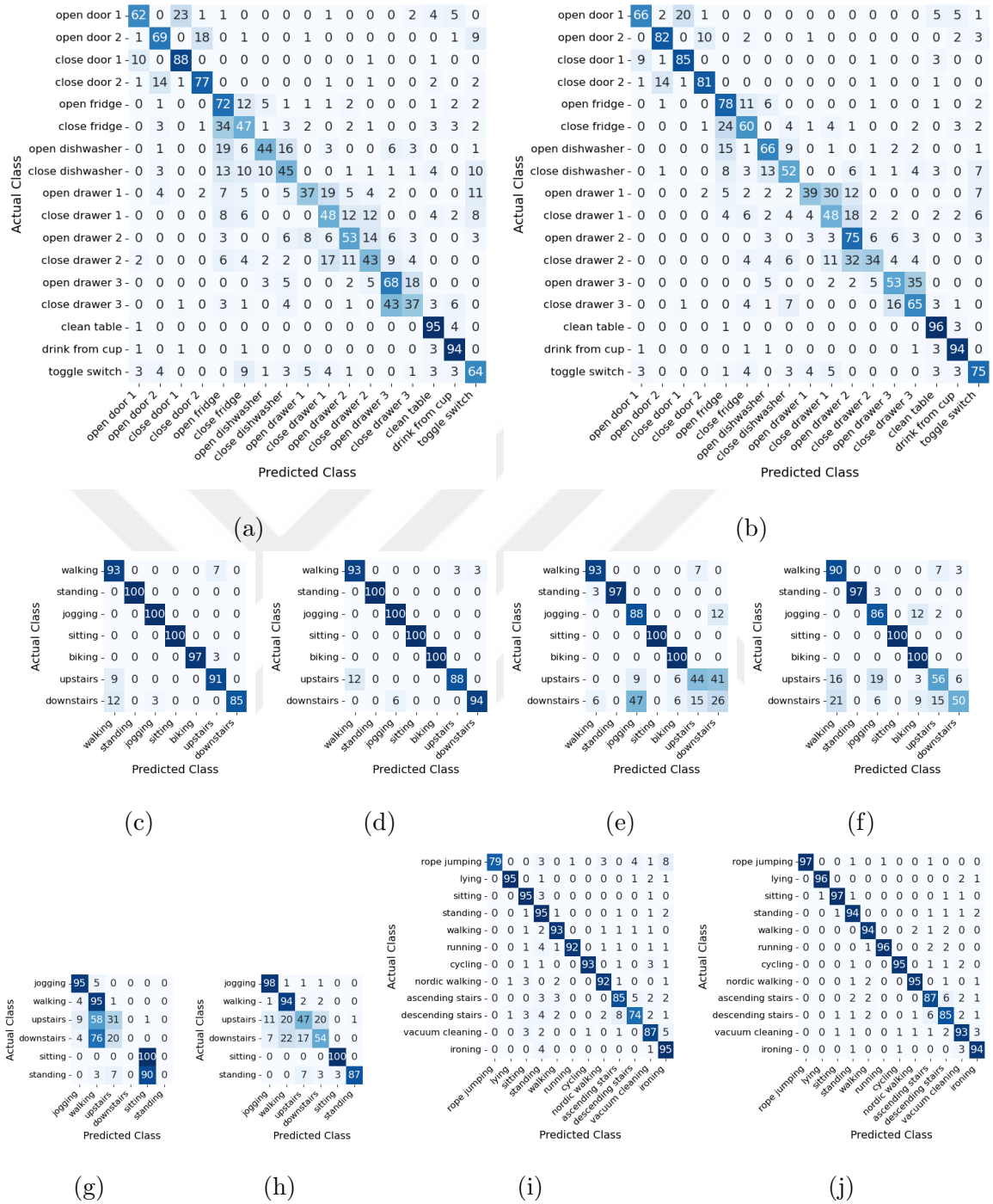


Figure 5.3. Confusion matrices. (a) LM and (b) LM-Att for Opportunity-DeepConvLSTM, (c) LM and (d) LM-Att for Sensors-DeepConvLSTM, (e) LM and (f) LM-Att for Sensors-SqueezeNet, (g) LM and (h) LM-Att for Wisdm-DeepConvLSTM, and (i) LM and (j) LM-Att for Pamap2-DeepConvLSTM.

For Opportunity-DeepConvLSTM (Figure 5.3a and 5.3b), using *LM*, we observe that 43% of the *close drawer 3* activity is predicted as *open drawer 3*. This, the highest confusion value, reduced to 16% with attention. Considering Sensors-DeepConvLSTM (Figure 5.3a and 5.3b), by using *LM*, the most confused activity is *downstairs walking* (12%), and this confusion is completely eliminated with the use of attention. Similarly, for Sensors-SqueezeNet (Figure 5.3e and 5.3f), the most confused activities are *downstairs* with *jogging*, *upstairs* with *downstairs*, and *downstairs* with *upstairs* (with 47%, 41%, and 15%, respectively). Using *LM-Att*, significant improvements are achieved in the first two confusions (reduced to %6 and %6, respectively), and the third remains constant. Moreover, for Wisdm-DeepConvLSTM (Figure 5.3g and 5.3h), *downstairs* is never correctly predicted with *LM* (0%), but with the use of attention, it increases to 54%. For Pamap2-DeepConvLSTM (Figure 5.3i and 5.3j), *ascending* and *descending stairs* are frequently confused with each other, and using attention, the confusion decreases by 2% (from 8% to 6%) in one case but also increases by 1% (from 5% to 6%) in another. However, the misprediction of *descending stairs*, the lowest-performing activity, decreases significantly and its performance increases from 74% to 85% using attention. Although low-performing activities benefit from attention, some new confusions may appear using attention. Nevertheless, these confusions are not high enough to decrease the overall performance of the *LM-Att*.

Analyzing all dataset-architecture pairs reveals that *LM-Att* tends to enhance the performance of poorly performing activities more significantly than it improves those already performing well. These enhancements occasionally surpass those achieved with *MM*, suggesting that the attention mechanism refines features to better learn challenging classes. While *MM* increases the model's overall learning capacity thanks to the increased model-size, similar or superior improvements can be attained by enhancing the model's learning capability through attention mechanisms.

5.4. Conclusion and Insights

This chapter investigated the impact of attention mechanisms on the performance and resource efficiency of different model sizes. The findings indicated that attention mechanisms, specifically CBAM, significantly enhance the performance of lightweight models across various datasets and metrics.

Attention mechanisms enabled lightweight models to achieve or surpass the performance of moderate-sized models while using significantly fewer resources. Consistent across multiple datasets, this integration enhanced feature extraction capabilities, making these models a viable and efficient alternative in resource-limited scenarios. By maintaining lower resource consumption and achieving high recognition performance, attention-enhanced lightweight models present a compelling solution for applications where efficiency is crucial.

Moreover, transitioning from lightweight to moderate models significantly increases computational requirements. However, using attention mechanisms improves performance with only a marginal increase in resource consumption, making it a practical solution for enhancing model efficiency. This makes *LM-Att* models particularly suitable for resource-constrained environments, where achieving high performance without incurring significant computational costs is crucial.

Additionally, the activity-based impact analysis reveals that *LM-Att* models tend to enhance the performance of poorly performing activities more significantly than those already performing well. This improvement is evident across various dataset-architecture pairs, with *LM-Att* occasionally surpassing *MM* in certain challenging activities such as *opening drawer 1*, *closing drawer 2*, and *rope jumping*. These findings suggest that the attention mechanism refines features to better learn challenging classes, offering performance enhancements comparable to those achieved by increasing the model's size.

In summary, attention mechanisms significantly enhance the feature extraction capabilities of lightweight models, enabling them to perform at levels comparable to larger models while maintaining lower resource consumption. These findings underscore the importance of balancing performance and resource efficiency in the design of HAR models, particularly for mobile and wearable devices with limited resources. Moreover, the performance improvements achieved with attention mechanisms could be further amplified by combining them with knowledge distillation techniques. Knowledge distillation involves training small models with the assistance of larger models, which can optimize both resource consumption and model performance. This combined approach could offer a powerful solution for deploying high-performing, resource-efficient HAR models in practical, real-world applications.

6. BOOSTING THE PERFORMANCE OF LIGHTWEIGHT MODELS WITH ATTENTION MECHANISM AND KNOWLEDGE DISTILLATION

In this chapter, we examine the synergistic effects of attention mechanisms and knowledge distillation on the performance of lightweight HAR models. Our main objective is to enhance these models' recognition accuracy while minimizing any increase in resource consumption, thus ensuring their feasibility for deployment on mobile and wearable devices.

Attention mechanisms, such as the previously investigated CBAM, have effectively highlighted the most informative features within the data, thereby improving model performance. Knowledge distillation, in contrast, involves transferring knowledge from a complex, high-performing teacher model to a simpler, more efficient student model. By combining these two techniques, we aim to develop lightweight models that achieve high recognition accuracy with minimal computational overhead. This chapter provides a detailed evaluation of these combined methods across various sensor-based HAR datasets, offering insights into their effectiveness and their potential to close the performance gap between lightweight models and their more complex counterparts. In particular, we investigate how attention mechanisms can be utilized to enhance the feature extraction process in the student models and how the distilled knowledge from the teacher models can further refine this process with different usages. We also examine the trade-offs between model complexity and performance, offering a detailed analysis of how these methods impact computational efficiency and accuracy. By emphasizing resource efficiency and performance, we aim to present a viable solution for deploying high-performing HAR models in real-world, resource-constrained environments.

6.1. Knowledge Distillation

The goal of Knowledge Distillation (KD) is to transfer knowledge from a complex model, referred to as the teacher network, to a less complex model, known as the student model. This is accomplished by progressively minimizing the difference between their output distributions. By following this approach, the student model can gradually mimic the teacher's predictions, providing essential insights in a condensed form. This method, also called response-based or logit-based knowledge distillation, is recognized for its simplicity and effectiveness in extracting knowledge from the teacher model [44].

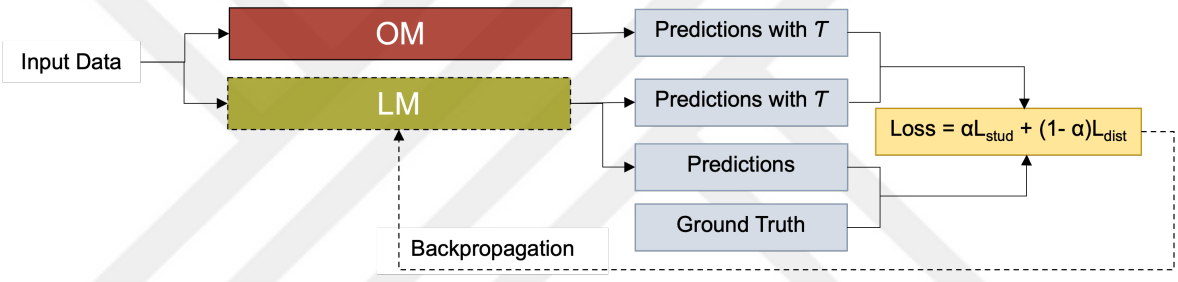


Figure 6.1. Response-based KD system.

In this study, response-based distillation is the fundamental KD strategy employed. To effectively train a lightweight model (LM), as depicted in Figure 6.1, we utilize response-based knowledge distillation. This process entails using a pre-trained original baseline model (OM) as the teacher model. The student model's loss is calculated by merging its inherent loss with the distillation loss derived from both teacher and student predictions. Since our problem involves multi-class classification, the student loss, L_{stud}, uses cross-entropy loss, and it is expressed as

$$L_{stud} = -\sum_{i} y_{true,i} \cdot \log\left(y_{student,i}\right). \tag{6.1}$$

The distillation loss, L_{dist}, employs KL divergence loss and it is computed as

$$L_{dist} = \tau^2 \sum_{i} \left(y_{teacher(\tau),i} \cdot \log\left(\frac{y_{teacher(\tau),i}}{y_{student(\tau),i}}\right)\right) \tag{6.2}$$

where temperature, τ, control the smoothness of the teacher's predictions. Therefore,

the overall loss of the system, $Loss$, can be written as

$$Loss = \alpha \cdot L_{stud} + (1 - \alpha) \cdot L_{dist} \tag{6.3}$$

where L_{stud} and L_{dist} are as defined before and α control the weight assigned to the student. The τ and α hyperparameters play a significant role in KD because they determine the softness of the teacher's predictions and the relative importance of the classification loss compared to the distillation loss. Proper tuning of these parameters is essential for effective knowledge transfer. A higher τ value results in a softer probability distribution from the teacher model, enabling the student model to learn more information from the distillation losses. Concerning the α value, a smaller α (less than 0.5) signifies a higher focus on the distillation loss during the student model's training process. While this can help the student learn from the teacher model, it may potentially prevent the student from completing the learning process on its own. Therefore, these parameters should be carefully fine-tuned based on the problem to strike an optimal balance between the student's independent learning process and the effective utilization of knowledge from the teacher. In KD experiments, the τ parameter is typically selected from the range of 0 to 20, while the α parameter is chosen from the range of 0 to 1. In our experiments, we explored τ values of 5, 10, and 15 and α values of 0.5, 0.7, and 0.9. Our objective is to create a lightweight model that achieves satisfactory performance while consuming fewer resources. To accomplish this, in our distillation experiments, we utilized the LM structure as the student model, with OM serving as the teacher model.

6.2. Combination of KD and Attention

After applying attention and distillation mechanisms separately, we then combine them to leverage their respective benefits through two distinct approaches. The first approach transfers attention from the teacher to the student via attention loss, effectively enhancing the response-based distillation (see Section 6.2.1). The second approach integrates the attention mechanism exclusively within the student model while simultaneously applying response-based distillation (see Section 6.2.2).

6.2.1. Response and Attention-based KD

We introduce attention mechanisms into our response-based distillation framework (refer to Figure 6.2) to examine how effectively attention-based distillation enhances the performance of lightweight HAR models. We employ both CA and/or SA-based distillation methods. The spatial attention maps for both the student and teacher have dimensions of $1 \times H \times W$, but their channel attention maps differ. Considering the DeepConvLSTM architecture, the teacher's channel attention map has dimensions of $64 \times 1 \times 1$, while the student's is $4 \times 1 \times 1$. Similarly, in SqueezeNet, the teacher's channel attention map in the first convolutional layer is $96 \times 1 \times 1$, whereas the student's is $6 \times 1 \times 1$. To address these mismatches in channel attention maps, we drew inspiration from [136] and FitNet [161], utilizing a 1×1 convolution to reduce the teacher's channel attention map to the same size as the student's in both cases. The overall loss function is then defined as

$$Loss = \alpha \cdot L_{stud} + (1 - \alpha) \cdot L_{dist} + \beta \cdot L_{att}. \tag{6.4}$$

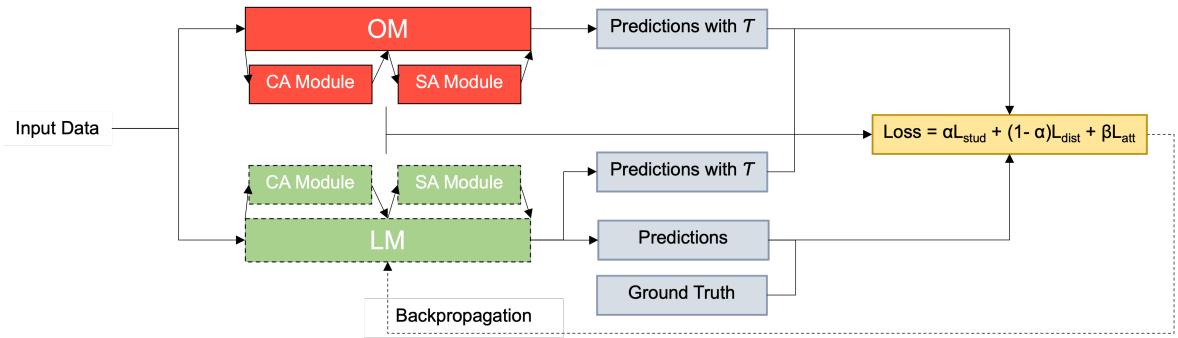


Figure 6.2. Response and attention-based KD (RAB-KD).

The key difference in the loss function is the incorporation of an attention-based term, L_{att}, which measures the mean squared error between the attention maps of the teacher and the student. Depending on the experimental setup, L_{att} may capture the differences in channel attention, spatial attention, or a combination of both. When both attention types are combined, the spatial attention output is used as a map, making the loss formulas for spatial and combined attention the same. Therefore, L_{att}

can be either L_{ca}, calculated as (*tea: teacher and stud: student*)

$$L_{ca} = \frac{1}{C}\sum_{k=1}^{C}\left(M_{c_tea}(k) - M_{c_stud}(k)\right)^2 \tag{6.5}$$

or L_{sa}, calculated as

$$L_{sa} = \frac{1}{H \times W}\sum_{i=1}^{H}\sum_{j=1}^{W}\left(M_{s_tea}(i,j) - M_{s_stud}(i,j)\right)^2. \tag{6.6}$$

6.2.2. Response-based KD with Attention Applied to the LM

Based on existing literature and our experimental results, we integrated attention mechanisms exclusively into the student model, excluding the teacher (see Figure 6.3). We chose this because we found that CBAM did not significantly improve the performance of the *OM*, but it did offer substantial gains for the *LM*. So, we focused on adding attention to the student model without distilling an attention map from the teacher. This approach eliminates the need for an *OM* trained with attention during the attention-distillation process. Furthermore, the loss function used remains consistent with that of response-based distillation.

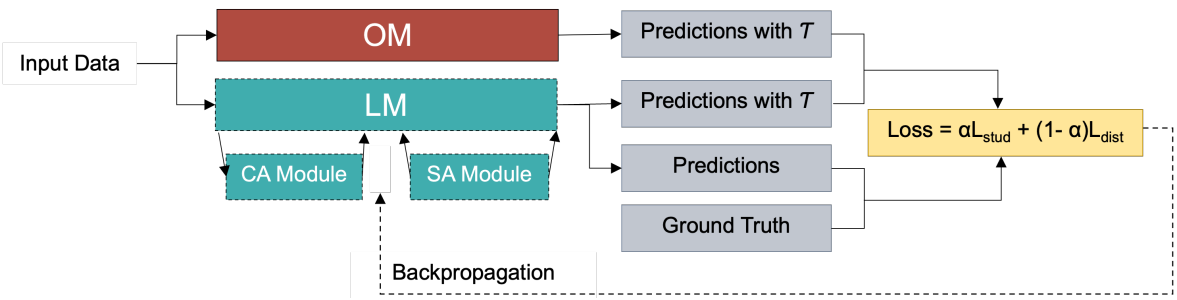


Figure 6.3. Response-based KD with attention applied to the student LM (RB-KD-Att).

6.3. Impact of KD versus Attention over the Lightweight Model

As discussed in Section 5.2, initial results indicate that increased model complexity and resource consumption generally enhance performance. Attention is one method to boost performance with minimal additional resource consumption. In this section,

we explore the impact of response-based KD on *LM* and compare its performance with *LM-Att* regarding resource usage and recognition performance to address RQ5.

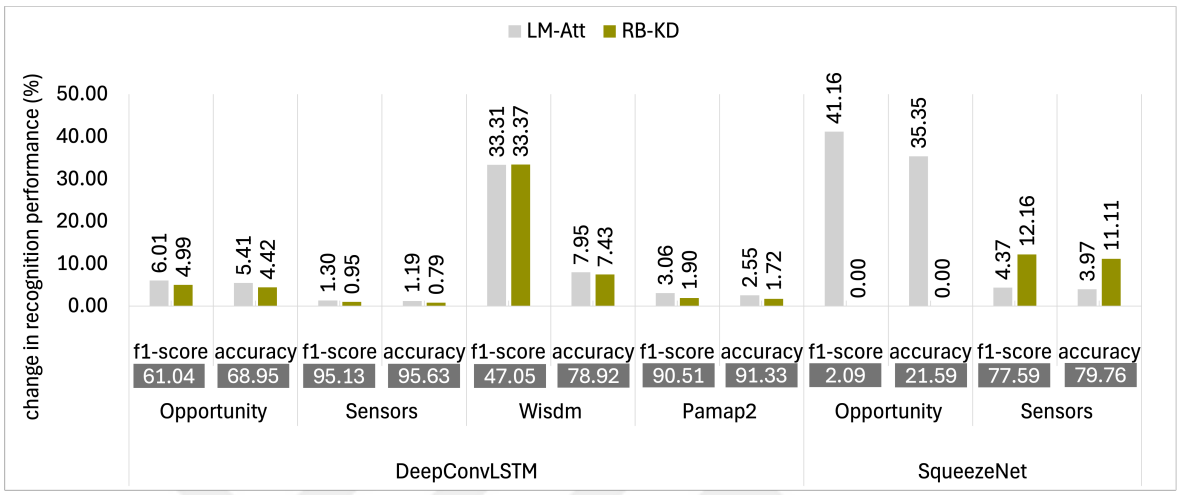


Figure 6.4. Changes in recognition performance for individual use of attention module (LM-Att) and response-based distilled LM (RB-KD) over baseline LM.

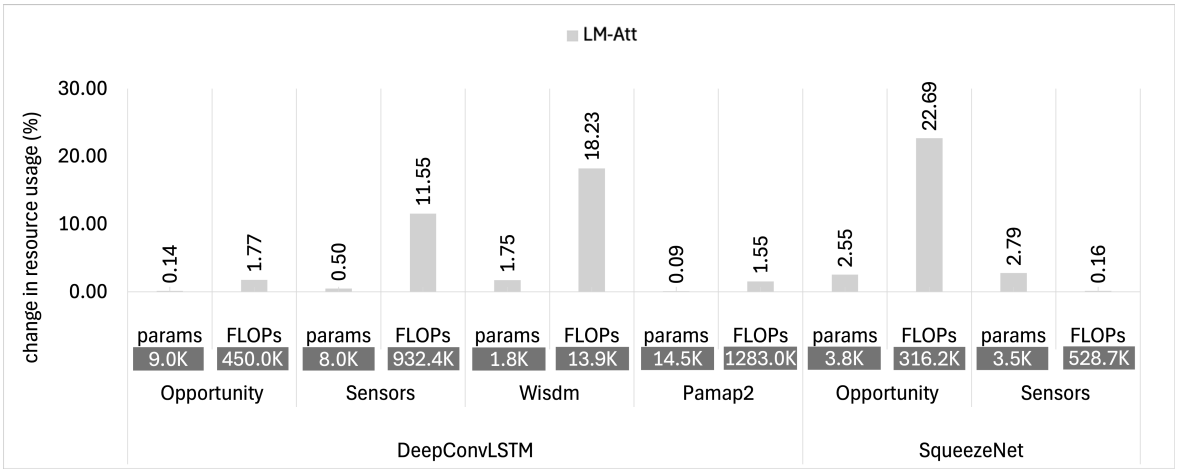


Figure 6.5. Changes in resource usage for individual use of attention module (LM-Att) and response-based distilled LM (RB-KD) over baseline LM.

In Figure 6.4, the differences in recognition performance, measured in terms of test f1-score and accuracy, are presented for the *LM* when only attention (grey bars)

and only KD (olive bars) are applied. The *LM*'s initial performance results are shown in the dark grey boxes. For instance, while the *LM* achieves an f1-score of 61.04% for the Opportunity dataset using DeepConvLSTM, this value increases by 6.01% with *LM-Att* and 4.99% with *RB-KD*. In Figure 6.5, the percentage increases in *LM*'s resource consumption, in terms of params and FLOPs, with the application of *LM-Att* (grey bar) are given. The baseline resource consumptions for *LM* are represented in the dark grey boxes. For example, the *LM* requires 9.04K parameters for the Opportunity using DeepConvLSTM (considered as 100%), and *LM-Att* increases this by 0.14% ($0.14 \times 9.04K$ params). Note that using only KD (*RB-KD*) does not increase the resource usage of *LM*. Due to the zero change, *RB-KD* results are not presented in the resource usage figure (Figure 6.5). When the gap between *OM* and *LM* is very large, as in the case of Opportunity-SqueezeNet (gap higher than 50%), the student has not been able to learn from the teacher. Besides, for Sensors-SqueezeNet, thanks to *RB-KD*, the performance has matched that of *OM*. For the rest of the pairs, where the gap is less than 15%, learning has occurred proportionally to the difference. More clearly, considering the DeepConvLSTM models, the gaps between *OM* and *LM*, in descending order, are 15.00% for Wisdm, 14.00% for Opportunity, 5.41% for Pamap2, and 3.57% for Sensors. The improvement achieved with *RB-KD* for the same order of datasets, respectively, is 7.43%, 4.42%, 1.72%, and 0.79% of test accuracy. Considering DeepConvLSTM, this suggests that as the gap between *OM*'s and *LM*'s predictions narrows, the student's ability to successfully distil knowledge from the teacher decreases, thus reducing the improvement rate.

Temperature (τ) and α are crucial hyperparameters in KD, influencing the teacher's softened predictions and balancing the significance of the classification loss against the distillation loss. Increasing the value of τ results in a softer probability distribution over classes, facilitating the extraction of more detailed information from the distillation losses. Thus, it is important to choose τ values that do not surpass the student's learning capacity. In four dataset-architecture pairs, the accuracy gap between the student and teacher models (OM and *LM*) exceeds 10%. For these cases, the optimal τ value is determined to be 5. On the other hand, for the two dataset-architecture

pairs where the difference is 5% or less, the best-performing τ values are 10 and 15. When the initial gap is large, employing a higher τ to enable more detailed learning might exceed the student's learning capacity, making lower τ values more effective.

Regarding the α value, selecting α values close to 1.0 emphasizes the student loss during the student's training process. For example, in the Wisdm using DeepConvLSTM, with $\alpha = 0.5$ and $\tau = 5$, the performance gap reduces to 7.57% using *RB-KD* (from a maximum gap of 15.00% achieved using *LM*), demonstrating effective utilization of the teacher's knowledge. In contrast, higher α values prioritize the student's classification loss with true labels, which can be beneficial when students' learning should be prioritized. This is evident in the SqueezeNet model on the Sensors dataset, where $\alpha = 0.9$ and $\tau = 5$ result in a minimal performance gap of 0.40% (from a maximum gap of 11.51%), underscoring the importance of the student's own learning process.

These hyperparameters should be carefully adjusted based on the problem's complexity to achieve an optimal balance between the student's independent learning process and the effective utilization of the teacher's knowledge. The Sensors dataset using DeepConvLSTM, with $\alpha = 0.9$ and $\tau = 10$, exhibit a small performance gap of 2.78%, indicating that the high τ allows the student model to effectively extract information from the teacher's softer predictions. Similarly, the DeepConvLSTM on Pamap2 with $\alpha = 0.7$ and $\tau = 15$ shows a moderate performance gap of 3.70%, suggesting that higher τ and α values can be advantageous when the student model has sufficient capacity to learn from the teacher.

Considering the results of *LM* with attention (*LM-Att*) in Figure 6.4 and Figure 6.5, the improvements achieved using *LM-Att* over the *LM* are already presented in Section 5.2. Here, we evaluate the impact of *LM-Att* compared to *RB-KD*. Upon analyzing the recognition performance of *LMs*, we observe that *LM-Att* consistently outperforms the other models for each of the four datasets in terms of recognition success using DeepConvLSTM. Specifically, *LM-Att* demonstrates superior performance,

followed by *RB-KD* and then *LM*. However, one exception exists for the Sensors dataset using SqueezeNet, where *RB-KD* outperforms *LM-Att* by 7.79% in f1-score and by 7.14% in accuracy.

While the influence of the distillation hyperparameters is quite evident, the effects of the attention hyperparameters are less clear. When examining the impact of parameters like the reduction ratio for CA, kernel size for SA, and the layer of application for convolution, it is challenging to determine their influence based on baseline performance, dataset complexity, or other factors. However, it is evident that CA has a more constrained impact compared to SA. This limitation likely arises because the *LM* has only 4 channels, resulting in CA producing just 4 attention values, which may hinder its ability to capture intricate patterns in the data, thereby affecting the abstraction of feature information. In contrast, spatial attention (SA) has a more pronounced effect on performance. The attention map generated by SA is more detailed and contains richer information, contributing to its positive impact. Additionally, the spatial attention module is more resource-efficient due to its lower parameter count compared to channel attention. This efficiency, combined with its ability to generate more informative attention maps, makes SA a more effective and practical choice in enhancing model performance.

6.4. Impact of Combined KD and Attention over the Lightweight Model

Our second phase continued to use the previous analysis, which indicated that most tasks performed better with attention mechanisms than distillation methods. We then conducted experiments on hybrid models that integrate both attention and distillation techniques (refer to Figure 6.2) to explore how their combined application can enhance recognition performance, addressing RQ6. Our investigation focused on a distillation strategy that emphasizes response and attention, aligning with existing frameworks in the literature. During these experiments, we carefully tested various configurations of attention and distillation, adjusting β values to identify the optimal balance for attention loss.

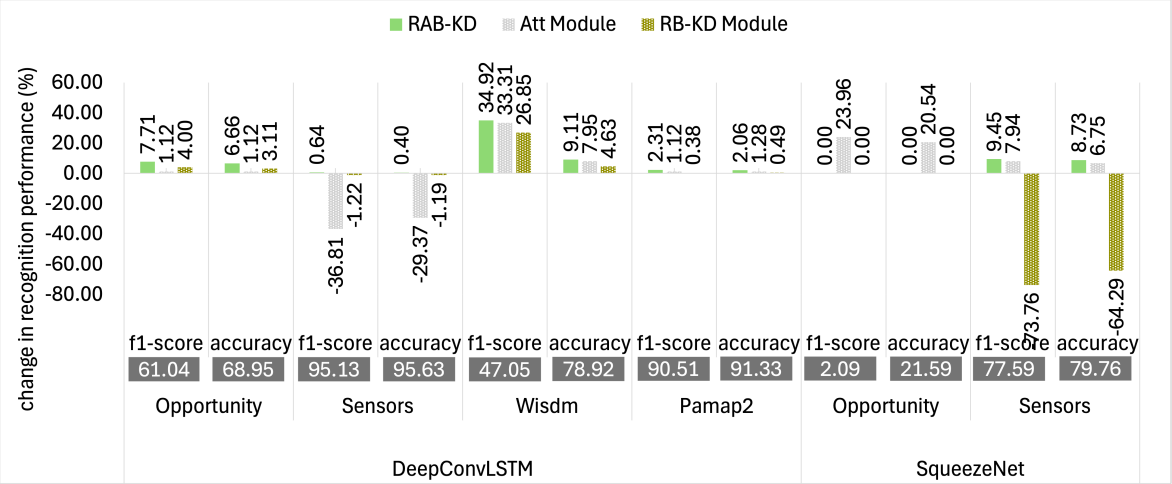


Figure 6.6. Changes in recognition performance for response and attention-based distillation on LM (RAB-KD) over baseline LM.

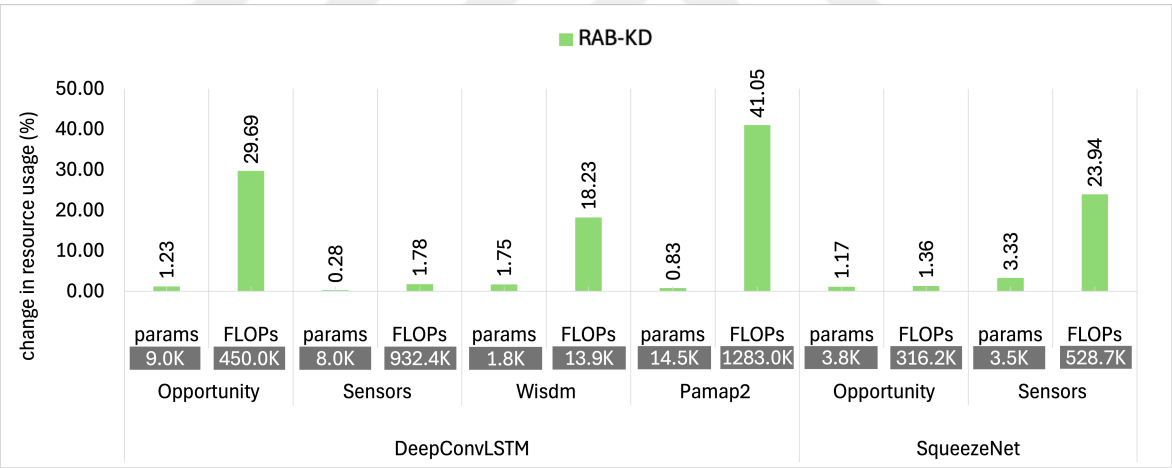


Figure 6.7. Changes in resource usage for response and attention-based distillation on LM (RAB-KD) over baseline LM.

The results of these experiments are presented in Figure 6.6 and Figure 6.7, which shows both the recognition performance and resource usages of models. In the figure, the top-performing response and attention-based distilled *LM* for each dataset is denoted as *RAB-KD*. We also analyzed the individual components of *RAB-KD*, specifically the attention module (denoted as *Att module* for related layer number, kernel size, and reduction ratio) and the distillation module (*RB-KD Module* for related

τ and α). It's important to note that using *RB-KD* alone does not impact the resource usage of *LM* (zero change), so it is not included in Figure 6.7 due to no change. Similarly, the resource usage for the *Att Module* is not shown as it is equal to that of *RAB-KD*.

In our experiments presented in Figure 6.6 and Figure 6.7, we observe that both *LM-Att* and *RB-KD* methods significantly improve the performance of *LM* for the Opportunity-DeepConvLSTM and Wisdm-DeepConvLSTM dataset-architecture pairs. For these pairs, using the *RAB-KD* method increases the f1-score and accuracy performance by 1% to 3%. However, for the Sensors-SqueezeNet combination, the performance of *RAB-KD* decreases compared to *RB-KD*. *RB-KD* nearly matches the teacher model's performance in test f1-score (*RB-KD* 89.75%, teacher *OM* 90.23% f1-score). Although *LM-Att* increases the *LM*'s f1-score from 77.59% to 81.96%, this value is still much lower than that of *RB-KD*. This suggests that while *LM-Att* improves performance, it may negatively affect the improvements made by distillation when combined.

RAB-KD negatively impacts performance only for the Opportunity-SqueezeNet combination, matching the initial *LM*'s f1-score and accuracy. Although *LM-Att* positively impacts performance, distillation does not contribute. From *OM-Att* experiments, we know that attention alone negatively impacts Opportunity-SqueezeNet, reducing the performance of *OM* by over 3%. Therefore, distilling attention from an already negatively affected *OM* may worsen performance. Besides, distillation alone is also In addition, distillation alone also poses a negative impact.

For the Sensors-SqueezeNet combination, the improvement provided by *RB-KD* in the *LM* is much higher than that provided by *LM-Att* (f1-score increases by 7.79%, accuracy by 7.14%). *RB-KD* nearly matches the performance of *OM* (approximately 0.5% lower f1-score). When adding the less effective attention mechanism to this well-performing distillation system, *RAB-KD*'s performance decreases by 2.4% in accuracy and 2.7% in f1-score compared to *RB-KD*. Finally, for Sensors-DeepConvLSTM and Pamap2-DeepConvLSTM, distilling attention does not bring any additional improve-

ments compared to using attention alone (*LM-Att*).

When we individually examine the distillation and attention modules of *RAB-KD* models (referred to as *RB-KD Module* and *Att Module* in Figure 6.6 and Figure 6.7), we find that for all dataset-architecture pairs, the combined approach, *RAB-KD*, surpasses the performance of using either *Att Module* or *RB-KD Module* alone, with the only exception being the *Att Module* for Opportunity-SqueezeNet. For most models, although each module alone does not significantly enhance performance, their combination results in substantial improvements. Additionally, these performance metrics surpass those of *MM* in 4 out of the 6 dataset-architecture pairs. The exceptions are Opportunity-DeepConvLSTM, which performs 1.05% below the *MM*, and Opportunity-SqueezeNet, which shows poor performance due to the use of distillation. Hence, it is evident that *RAB-KD* models can achieve performance levels comparable to scaling up the *LM* model to the *MM* model, but with 2.1-3.7x fewer parameters and FLOPs.

In summary, we observe that when the *OM* is negatively affected by attention, using *RAB-KD* may not be reasonable. Besides, it might not be beneficial if there is a large gap between the performance of *LM-Att* and *RB-KD*. However, if both *LM-Att* and *RB-KD* contribute positively and their improvements are not significantly different from each other, using *RAB-KD* can be advantageous.

In KD, the teacher model is usually a large-scale, pre-trained model whose pre-trained version is used to guide the student model's learning during training. This setup is suitable for response-based KD. However, when attention-based distillation is desired, having the same attention mechanism in both the teacher and student models requires using an attention-based teacher model, though this may not always be possible. Besides, we know that the contribution of attention is not substantial in large models (*OM*) as investigated in Chapter 4, which already achieve high recognition performance. In contrast, the effect of attention on the *LM* is substantial, as evident in Chapter 5. It is, therefore, worthwhile to investigate whether incorporating attention directly into the student model, rather than through the teacher, can yield

comparable performance gains to *RAB-KD*. Further experiments were conducted using the response-based knowledge distillation (*RB-KD*) setup presented in Figure 6.3, with attention applied to the student to answer RQ7. This approach addresses the challenge of finding an attention-based teacher model and explores the possibility of achieving comparable performance by focusing attention solely on the student model.

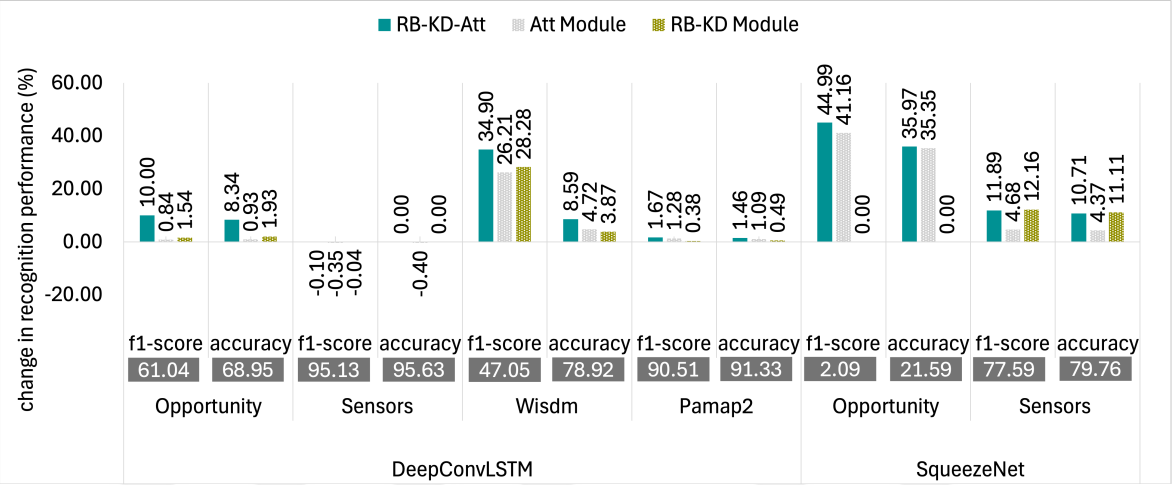


Figure 6.8. Changes in recognition performance for response-based distillation on LM with attention (RB-KD-Att) over baseline LM.

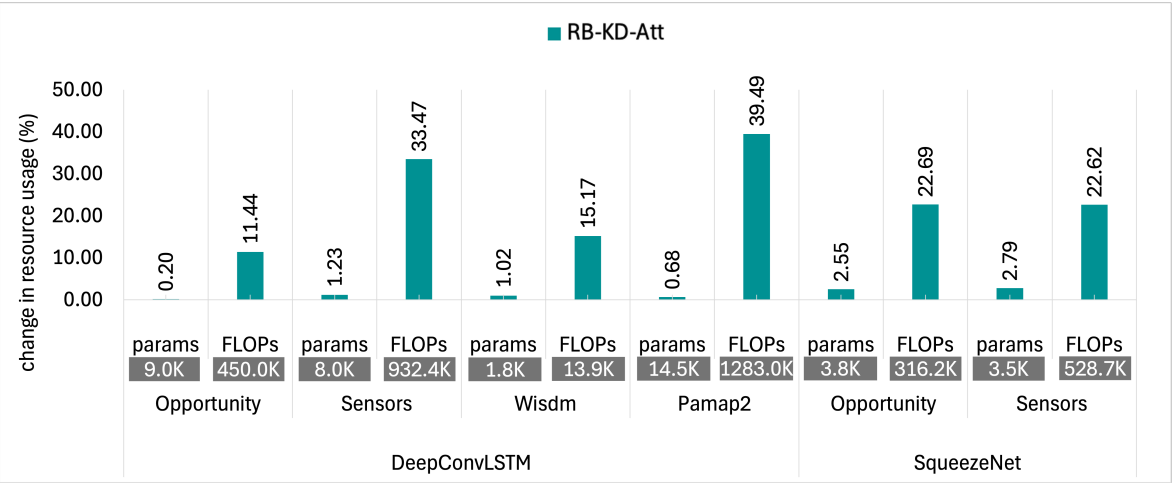


Figure 6.9. Changes in resource usage for response-based distillation on LM with attention (RB-KD-Att) over baseline LM.

Teacher models in KD are usually large-scale, pre-trained models that guide student models' learning during training. Response-based KD can be performed with this setup. However, implementing attention-based distillation necessitates that both the teacher and student models employ the same attention mechanism, which may not always be possible. Furthermore, as discussed in Chapter 4, we know that attention does not significantly enhance large models (*OM*) that already exhibit high recognition performance. In contrast, Chapter 5 demonstrates that attention significantly impacts the *LM*. Thus, it is valuable to explore if integrating attention directly into the student model, instead of through the teacher, can achieve similar performance improvements as *RAB-KD*. Additional experiments are performed using the response-based knowledge distillation (*RB-KD*) setup shown in Figure 6.3, applying attention to the student model. This method overcomes the difficulty of needing an attention-based teacher model and examines the potential for attaining comparable performance by concentrating attention solely on the student model.

The results are displayed in Figure 6.8 and Figure 6.9. Similar to the presentation in Figure 6.6 and Figure 6.7, the *RB-KD-Att* model is shown along with the performances of its individual modules. Additionally, the resource usages of the *RB-KD Module* and *Att Module* are not included, as there is no change in resource usage compared to the *LM* and they share the same resource usage as the *RB-KD-Att Module*.

Individual modules of the best-performing *RB-KD-Att* models generally perform lower, except Sensors-SqueezeNet. Compared to *RAB-KD*, the performance of *RB-KD-Att* slightly decreases recognition performance for three dataset-architecture pairs (Sensors-DeepConvLSTM, Wisdm-DeepConvLSTM, and Pamap2-DeepConvLSTM) by 0.03% to 0.75% in f1-score and by 0.40% to 0.60% in accuracy. Overall, this represents a performance decrease of less than 1%. However, for the Opportunity-DeepConvLSTM, *RB-KD-Att* increases the performance of the best-performing *RAB-KD* by 2.29% in f1-score and 1.68% in accuracy. For the Opportunity-SqueezeNet, *RB-KD-Att* not only surpasses *RAB-KD* but also outperforms the previously best-performing method, *LM-Att*. Similarly, for Sensors-SqueezeNet, *RB-KD-Att* exceeds the performance of both

RAB-KD and the previously best-performing method, *RB-KD*. Overall, *RB-KD-Att* achieves the best performance for four dataset-architecture pairs and performs very close to the best (within 1%) for the remaining two pairs.

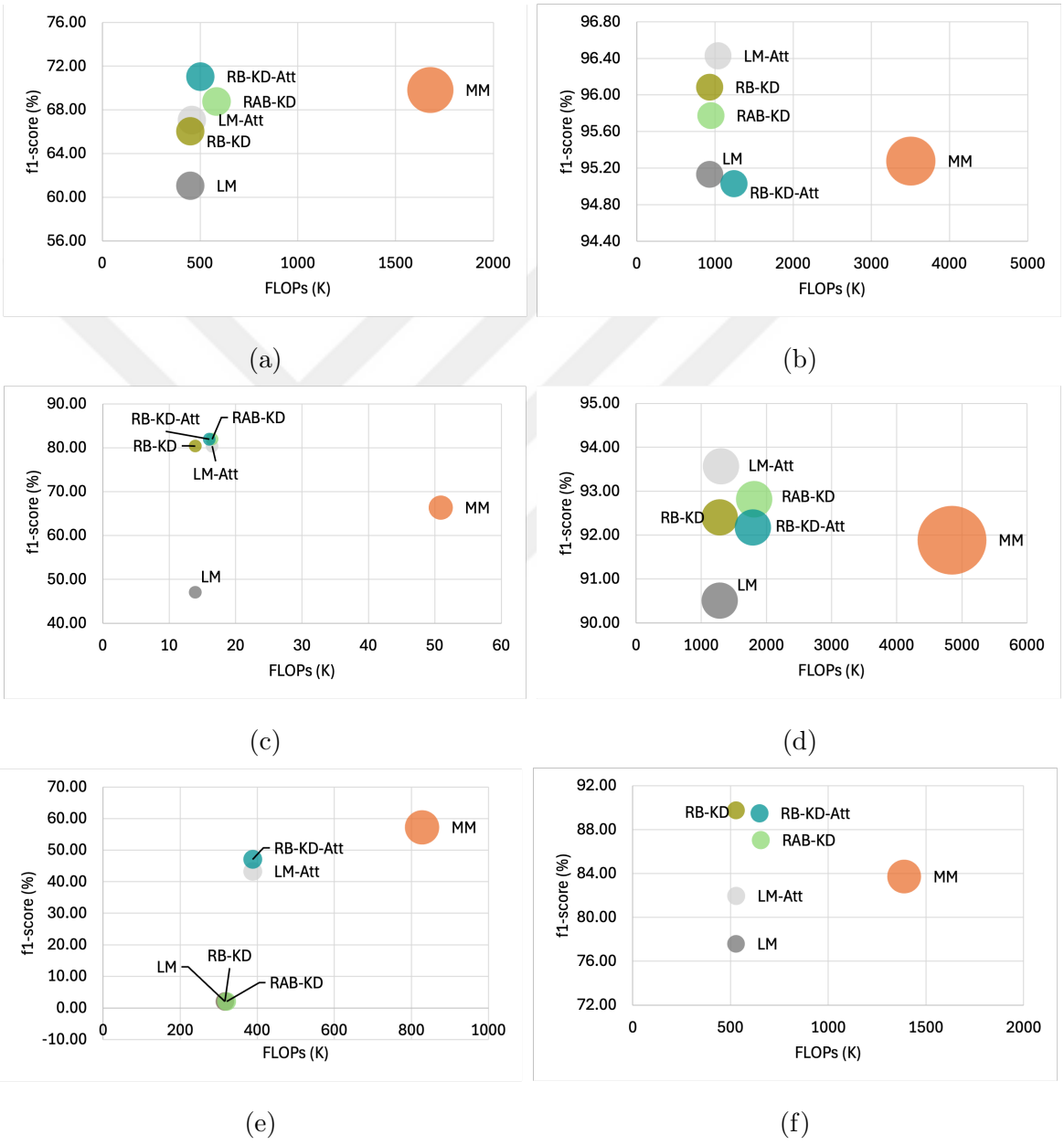


Figure 6.10. Ball chart reporting the recognition performance vs. resource consumption of LM, boosted LMs and MM. (a) Opportunity-DeepConvLSTM, (b) Sensors-DeepConvLSTM, (c) Wisdm-DeepConvLSTM, (d) Pamap2-DeepConvLSTM, (e) Opportunity-SqueezeNet, (f) Sensors-SqueezeNet.

In Figure 6.10, we provide a summary of the performance improvements and resource savings achieved by the proposed *LM* using KD and attention, compared to *MM*, across six dataset-architecture pairs. In this figure, the x-axis indicates the test f1-score, the y-axis shows the FLOPs, and the size of the circles corresponds to the number of model parameters. It is evident that, while the performance of the *LM* alone is lower than that of the *MM* across all pairs, the enhanced *LM* surpasses *MM* in all pairs except for Opportunity-SqueezeNet.

Among the five pairs where performance surpasses *MM*, three show an f1-score improvement of 1% to 2%, one shows an improvement of approximately 6%, and another shows an improvement of approximately 16%. These models, which outperform the *MM*, consume 2.6x-3.7x fewer FLOPs and parameters compared to *MM*. Thus, while *MM* achieves better performance over *LM* with a 2.6x-3.7x increase in resource consumption, the boosted *LM* variants not only surpass *MM* in recognition performance but also achieve this with minimal increases in model complexity.

6.5. Conclusion and Insights

In the previous chapter, we observe that attention mechanisms alone improve the performance of the lightweight models. Knowledge distillation, specifically response-based knowledge distillation, proves to be an effective strategy for transferring knowledge from a complex teacher model to a simpler student model. This process allows the student model to mimic the teacher's predictions, thereby enhancing its recognition accuracy. Consequently, we apply response-based knowledge distillation. When response-based knowledge distillation is applied, all experiments, except for a model where the difference between the student and teacher exceeds 50% accuracy, demonstrate performance improvements for the *LM*. Notably, for one dataset-architecture pair, Sensors-SqueezeNet, the performance of the *LM* approaches the performance of the original model. Additionally, this improvement does not result in any increase in the resource consumption of the *LM*.

To combine the advantages of both attention and distillation, attention-based KD (*RAB-KD*) is investigated. This approach yields substantial performance gains by integrating the strengths of both techniques. Hybrid models that incorporate both attention and KD outperform those utilizing either method alone, demonstrating marked improvements in recognition accuracy and resource efficiency across various dataset-architecture pairs.

Given the potential unavailability of attention-based teacher models and recognizing that attention does not significantly enhance the performance of larger models, experiments are conducted by applying attention mechanisms directly to the student model (*RB-KD-Att*). This structure allows the benefits of attention mechanisms without relying on attention-based teacher models. The results from these experiments are promising, indicating that for five out of six dataset-architecture pairs, the *LM* surpasses the performance of moderate-sized models while consuming significantly 2.6x-3.7x fewer resources (FLOPs and params).

In conclusion, the integration of attention mechanisms and knowledge distillation presents a potent approach to enhancing the performance of lightweight HAR models. Attention mechanisms alone can significantly boost the performance of lightweight models, and their combination with KD results in even more substantial improvements. This combined strategy proves highly effective for deploying high-performing, resource-efficient models in real-world applications, particularly on mobile and wearable devices. By balancing performance gains with resource efficiency, this approach holds significant promise for practical, resource-constrained environments.

7. COMPARING THE PERFORMANCE OF BOOSTED LIGHTWEIGHT MODELS WITH COMPRESSED MODELS USING QUANTIZATION AND PRUNING

This chapter focuses on several compression methods to enhance the sensor-based HAR models. Our goal is to reduce computational and memory demands while maintaining recognition performance. Among the commonly used compression techniques, including model pruning, parameter quantization, knowledge distillation, lightweight model design, and low-rank decomposition [95], model pruning and parameter quantization are two of the most widely utilized techniques in practice [162, 163]. Consequently, in this chapter, we evaluate the comparative effectiveness of boosted LMs (with attention and KD) and conventional pruning and quantization techniques, both from a recognition success and resource-efficiency perspective. Quantization reduces the model size and computational load by decreasing the precision of model parameters, while pruning removes redundant weights to simplify the models.

We compare the performance of these compressed models with our boosted *LMs* (*LM-Att*, *RAB-KD* and *RB-KD-Att*) to evaluate their efficiency in maintaining high recognition accuracy with lower resource usage. In the rest of the chapter, we use the same datasets and deep models that we presented in Chapter 3 and subsequently investigated in Chapter 4, 5 and 6. The insights from this chapter's analysis provide practical guidance for deploying HAR models in resource-limited environments.

7.1. Model Compression with TensorFlow Lite

TensorFlow Lite [33] is an open-source framework, and it is specifically designed to enhance the efficiency of running deep learning models on devices with limited computational resources like mobile phones, edge devices, and microcontrollers. To optimize original deep learning models, we first convert them to their lite versions using TensorFlow Lite. The conversion process involves transforming the model from

its original TensorFlow format to the TensorFlow Lite format, optimizing it for both size and speed. TensorFlow Lite converter facilitates this transformation by generating a *.tflite* file extension from the pre-trained model, which can then be deployed on various devices for inference. Therefore, we first converted our TensorFlow models to their TensorFlow Lite formats, denoted as *Lite*.

In addition to the basic conversion, TensorFlow Lite provides additional tools for further optimizing the models. Two key techniques included in TensorFlow Lite are pruning and quantization. These techniques significantly enhance the efficiency of models, making them more suitable for devices with limited resources. After converting our TensorFlow models to their *Lite* versions, we applied these optimization techniques to enhance their performance further. The implementation of quantization methods with TensorFlow Lite consists of converting the model to its *Lite* version and applying TensorFlow Lite's optimization tools to quantize the model. For pruning, the process is similar (*Lite* version conversion and TensorFlow Lite's pruning tool to optimize), with an additional step of training to fine-tune the remaining weights, with the aim of achieving optimal performance before deployment.

7.2. Parameter Quantization

Quantization reduces the number of bits used to represent model parameters, thereby decreasing model size and computational requirements. In this part of the thesis, we utilize two quantization methods: Float16 Quantization (FQ) and Dynamic Range Quantization (DRQ), addressing RQ8.

FQ reduces precision from 32-bit to 16-bit floating points, effectively decreasing the model size and speeding up inference time. This method balances reducing model size and preserving accuracy. It is particularly effective on hardware that supports 16-bit floating-point operations, such as certain specialized hardware accelerators. During inference on CPUs, weights are by default converted back to 32-bit format, while on GPUs, computations can be directly performed using the reduced precision values.

DRQ reduces the size of deep learning models by converting floating-point weights into 8-bit representations, which use approximately a quarter of the original memory space. This significant reduction in model size is especially beneficial for deployment on devices with limited resources. However, *DRQ* primarily focuses on optimizing memory usage rather than drastically improving computational speed. During inference, *DRQ* dynamically quantizes activations based on their range and uses 8-bit weights and activations for computations, with outputs still in floating-point format. Thus, the speed improvements are somewhat limited due to the persistence of some floating-point operations. This method ensures efficient memory use while maintaining reasonable computational performance, making it ideal for devices with limited computational power and stringent memory constraints.

Overall, both *FQ* and *DRQ* are crucial for deploying deep learning models on resource-constrained devices, enhancing memory efficiency while maintaining performance. *FQ* offers a balanced approach by decreasing model size and maintaining high accuracy, particularly on 16-bit compatible hardware. *DRQ* excels in significantly reducing model size, making it ideal for memory-limited environments, with a moderate impact on computational speed.

In the experiments, pre-trained *OM* models for each of the six architecture-dataset pairs, such as Opportunity-DeepConvLSTM, are converted to their TensorFlow Lite versions as described in Section 7.1. For both *FQ* and *DRQ* conversions, once the *OM* models are quantized using TensorFlow Lite, no further re-training is necessary. The results of the quantization are shown in Figure 7.1, 7.2 and 7.3. We first discuss the results regarding recognition accuracy, as presented in Figure 7.1 (where the x-axis represents validation accuracy). Following this, we examine the resource usage metrics, specifically inference time and model size, shown in Figures 7.2 (with the x-axis indicating inference time) and 7.3 (with the x-axis indicating model size), respectively. In these figures, the y-axis represents the architecture-dataset pairs, while the bars show the results for the converted *FQ* and *DRQ* models, along with the previously created *Lite* and *OM* versions for the corresponding architecture-dataset pairs. Finally, we

summarize the insights gained from this part of the study, focusing on the trade-offs between accuracy, computational efficiency, and model size.

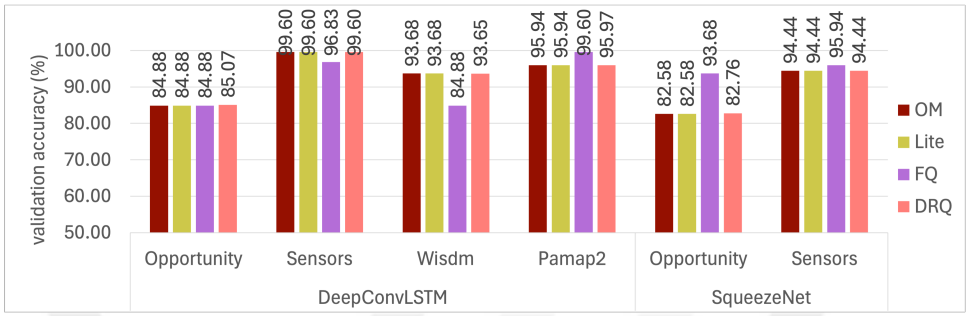


Figure 7.1. Validation accuracy (%) of quantized models.

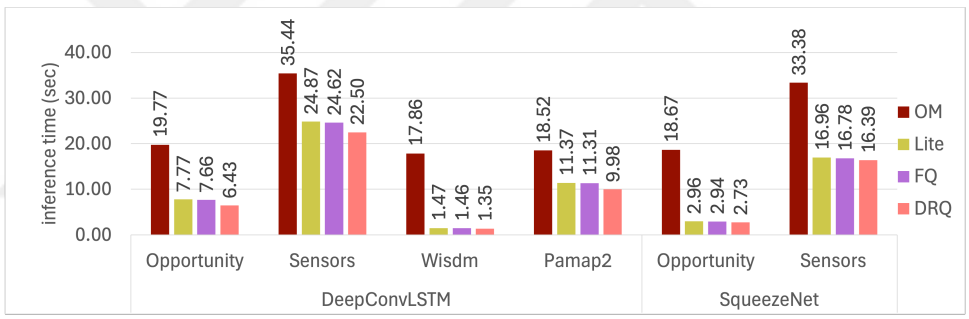


Figure 7.2. Inference time (sec) of quantized models.

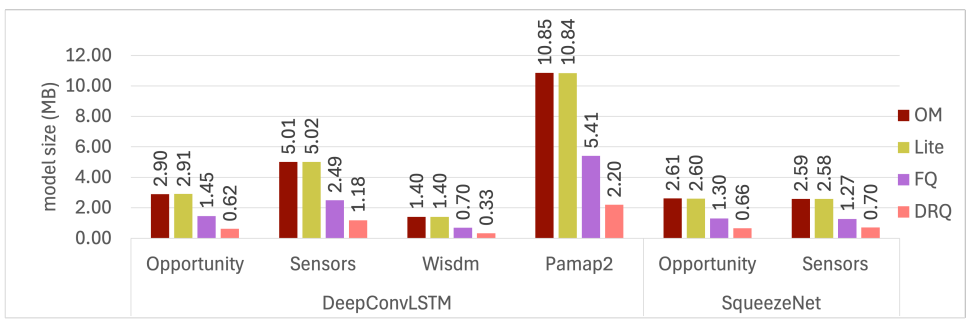


Figure 7.3. Model size (MB) of quantized models.

For validation accuracy (Figure 7.1), the *Lite* versions of the models achieve identical performance compared to the *OM*. Notably, using *DRQ*, the recognition performance of the *Sensors* dataset remains unchanged using both DeepConvLSTM and

SqueezeNet architectures. Additionally, there is a slight increase of 0.19%, 0.19%, and 0.03% for the Opportunity-DeepConvLSTM, Opportunity-SqueezeNet, and Pamap2-DeepConvLSTM pairs, respectively. However, a minor decrease of 0.03% is observed for the Wisdm-DeepConvLSTM pair.

In contrast, when *FQ* is used, the performance remains unchanged only for the Opportunity-DeepConvLSTM pair. Other dataset-architecture pairs experience significant increases or decreases in terms of accuracy. For instance, the performance of the Opportunity-SqueezeNet pair improves by 11.10%, rising from 82.58% to 93.68% compared to the *OM*. Conversely, the Wisdm-DeepConvLSTM pair achieves a performance drop from 93.68% to 84.88% with *FQ*. This indicates that while *DRQ* maintains consistent performance, *FQ* can lead to significant performance variations across different dataset-architecture pairs.

Examining inference time usage (Figure 7.2), transitioning from the *OM* model to TensorFlow Lite models (*Lite*, *FQ*, and *DRQ*) reduces inference time by a factor ranging from 1.44 to 12.21. However, the inference time usage of *FQ* models is almost identical to that of *Lite* across all dataset-architecture pairs. In contrast, *DRQ* models require 1.03x to 1.21x less inference time compared to *Lite* models. This indicates that compressed models (*FQ* and *DRQ*) consume nearly the same inference time as *Lite* models.

In Figure 7.3, the impact of reducing precision on model size through quantization is presented. Transitioning from the *OM* model to the *Lite* model slightly reduces the model size (decrease up to 0.01MB), except Opportunity-DeepConvLSTM (increase of 0.01MB). Applying *FQ* shrinks the *Lite* model size by a factor of 2, maintaining this reduction consistently across all dataset-architecture pairs. In contrast, the model size reduction achieved with *DRQ* ranges between 3.7x and 4.9x that of the *Lite* model. This indicates that, in terms of model size, the most advantageous quantization methods are *DRQ*, *FQ*, and *Lite*, in that order. This is not surprising, as *DRQ* reduces the precision from 32-bit to 8-bit, while *FQ* reduces it to 16-bit, resulting in more

significant size reductions with *DRQ*.

In summary, *DRQ* stands out as the most effective method for reducing model size, making it ideal for memory-constrained environments while maintaining consistent performance across various dataset-architecture pairs. *FQ* provides a balanced approach, achieving substantial size reductions and comparable inference time usage to *Lite*, with some variations in accuracy depending on the dataset-architecture pair. The *Lite* models, although not as compact as *DRQ* or *FQ*, offer significant improvements in inference time and model size compared to the *OM*. Thus, the choice of quantization method should be based on the specific requirements and constraints of the deployment environment, balancing memory efficiency, computational load, and performance accuracy.

7.3. Model Pruning

Pruning is a powerful technique aimed at optimizing deep learning models by reducing their computational and memory requirements. By eliminating less significant weights, such as those with low magnitudes, models become simpler and more efficient without significantly compromising performance. This technique is particularly valuable in applications such as sensor-based HAR, where resource efficiency is crucial. In this section, we explore two prominent pruning strategies: Constant Pruning (*CP*) and Polynomial Decay Pruning (*PDP*), evaluated across various pruning ratios from 10% to 99% to address RQ9.

CP involves removing a fixed percentage of redundant weights at each pruning step, maintaining a consistent pruning rate throughout the process. This method effectively reduces model complexity uniformly, enhancing simplicity and predictability. The primary advantage of *CP* is its straightforward implementation, as the pruning rate remains unchanged. A crucial component in *CP* is the pruning schedule, which takes the training step as input and outputs a sparsity percentage. This percentage determines which weights to prune based on their magnitude, ensuring a systematic

and effective pruning process. The constant rate allows for steady model size reduction, making it easier to manage and deploy in resource-constrained environments.

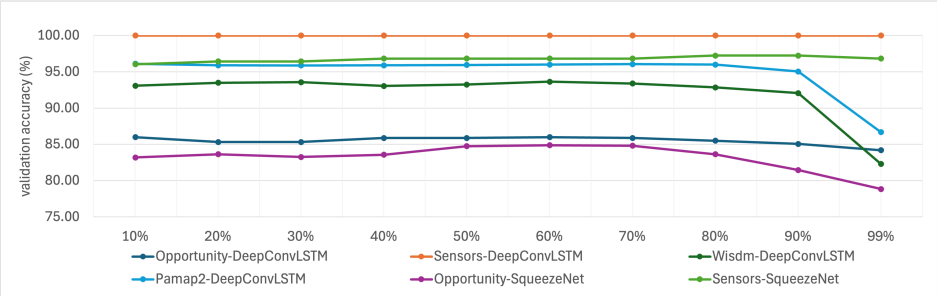


Figure 7.4. Validation accuracy (%) of CP method across different pruning rates.

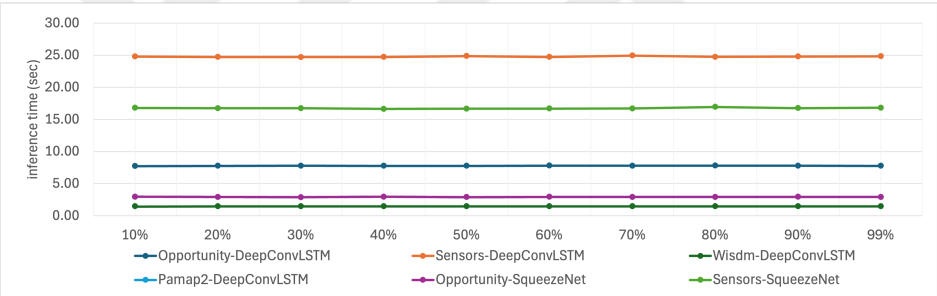


Figure 7.5. Inference time (sec) of CP method across different pruning rates.

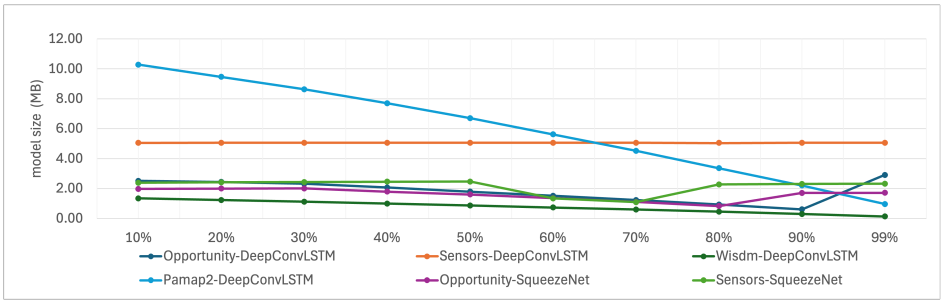


Figure 7.6. Model size (MB) of CP method across different pruning rates.

PDP starts with a lower predefined pruning rate that gradually increases over time. This approach allows the model to retain more information during the initial stages of training and fine-tune weights more effectively as training progresses. The

adaptive nature of *PDP* potentially leads to better performance compared to a constant pruning rate. The pruning schedule for *PDP* is crucial as it ensures that the model begins with intensive pruning, which gradually reduces, allowing for more refined weight adjustments in the later stages of training. This method enables the model to adapt more flexibly, preserving essential features and focusing on refinement as training progresses. By gradually increasing the pruning rate, *PDP* allows for a balance between pruning and learning, which can lead to improved model performance and generalization.

Both *CP* and *PDP* offer distinct advantages for model pruning. *CP* provides simplicity and uniformity, making it easy to implement and predict. It is particularly useful for applications where a steady reduction in model complexity is desired. On the other hand, *PDP* offers an adaptive and refined approach, starting with aggressive pruning that gradually decreases. This method is beneficial for applications requiring a more flexible and fine-tuned pruning strategy. Implementing these techniques enhances model efficiency, making them well-suited for performance-optimized applications.

Analyzing the impact of different pruning rates on validation accuracy reveals notable findings, as presented for *CP* in Figure 7.4, 7.5 and 7.6. Note that we provide more detailed results for *CP* as it is the most direct method to observe the impact of pruning. However, we also include results for *PDP* in the text. In Figure 7.4, 7.5 and 7.6, the x-axis presents pruning rates ranging from 10% to 99%, indicating the percentage of parameters removed from the model, while the y-axis presents recognition performances and resource usage. For the *PDP* method, the effect on validation accuracy remains under 1% for four out of six dataset-architecture pairs. However, for Opportunity and Sensors using SqueezeNet, this difference increases to 1.74% and 3.57%, respectively. Conversely, the *CP* method, which employs a fixed pruning rate, shows a general decline in validation accuracy beyond a 60% pruning rate. Up to this point, the accuracy remains nearly stable. This trend is evident in the analysis presented in Figure 7.4. Specifically, increasing the pruning rate from 90% to 99% for Wisdm-DeepConvLSTM and Pamap2-DeepConvLSTM results in performance

drops from 92.05% to 82.29% and 95.03% to 86.68%, respectively. Additionally, for Opportunity-SqueezeNet, there is a significant performance drop after the 70% pruning rate, with decreases of 1.18% from 70% to 80%, 2.18% from 80% to 90%, and 2.61% from 90% to 99%. Results show that *CP* maintains stable performance up to a specific pruning rate but begins to decrease rapidly beyond that threshold. This drop is typically observed after a pruning rate of 70% in our dataset-architecture pairs. However, the Sensors dataset shows consistent performance regardless of the pruning rate, maintaining a near-perfect accuracy of 100% even when pruned to 99%. This indicates that the model can handle high pruning rates without compromising accuracy. This stability is more pronounced in DeepConvLSTM than in SqueezeNet, where the latter exhibits more dramatic performance decreases beyond the 60-70% pruning rate.

These observations suggest that the impact of pruning is more dependent on the dataset characteristics than on the specific architecture used. The consistent performance of the Sensors dataset across both DeepConvLSTM and SqueezeNet architectures indicates that some datasets are inherently more robust to pruning. On the other hand, datasets like Opportunity show significant performance drops beyond certain pruning thresholds, highlighting the importance of dataset-specific characteristics in determining the effectiveness of pruning. Considering dataset-specific characteristics, as mentioned in Section 3.2, Opportunity dataset includes 17 activities that are more likely to be confused with each other, (such as *open door 1* and *close door 1*), compared to the activities in the other three datasets. As a result, pruning begins to affect the performance earlier, in the 60%-70% range. In contrast, the Sensors dataset has fewer activities, 7 in total, and a perfectly balanced class distribution. From this perspective, it poses fewer challenges and is almost unaffected by pruning. Pamap2 and Wisdm datasets fall in between in terms of difficulty (considering class distribution and the number of activities), and we observe that pruning starts to impact two datasets beyond the 70%-80% range.

When examining the resource consumption metrics, particularly the inference time, we observe that there is almost no variation concerning the pruning rate for

both *CP* and *PDP* methods. The reason for this is that unused parameters, which are considered zero, are still included in the computations with the same precision during the inference phase. Since this study does not focus on a system where zero-valued parameters are excluded from the calculations, there is no significant advantage or disadvantage in inference time related to the pruning rate. This is illustrated for *CP* in Figure 7.5, and a similar situation is observed for *PDP* as well. However, when investigating model size, as mentioned before, we apply a standard compression algorithm, gzip, to better investigate the compression benefits of pruning. The model size results of *CP* for different pruning sizes are presented in Figure 7.6. As mentioned previously, we only present results for *CP* since this is the most direct method to observe the impact of pruning on model size.

We observe that higher pruning rates generally lead to significant reductions in model size. For example, Opportunity-DeepConvLSTM's model size decreases substantially from 2.52MB at 10% pruning to 0.61MB at 90% pruning. This trend demonstrates that pruning effectively reduces the memory footprint of models, which is crucial for deployment on resource-constrained devices. However, the extent of size reduction can vary across different datasets, even when using the same architecture and pruning method. For instance, Sensors-DeepConvLSTM shows a constant model size of 5.06MB regardless of the pruning rate. This suggests that the impact of pruning on model size might be influenced by the specific characteristics and distributions within each dataset, similar to the validation accuracy observations presented in Figure 7.4.

In some cases, we observe a slight increase in model size at very high pruning rates, as seen with Opportunity-SqueezeNet and Sensors-SqueezeNet, which slightly increase at 99% pruning. This suggests that extremely high pruning rates might lead to inefficiencies or redundancies that offset some of the benefits of pruning. These inefficiencies could be due to how pruned parameters are handled or the remaining structure of the model. Overall, these results highlight that while pruning is generally beneficial for reducing model size, the effectiveness can vary, and extreme pruning may not always lead to optimal outcomes. Thus, careful consideration of the pruning rate

is essential to balance between memory savings and potential inefficiencies.

After investigating the impact of the pruning rate on recognition success and resource consumption of HAR models, we then compare the performance of pruned models with *OM* and *Lite* models for each dataset-architecture pair. We selected to present results for the 60% pruning rate since this value is considered an acceptable threshold that minimizes resource consumption without significantly compromising performance. The results are presented in Figure 7.7, 7.8 and 7.9.

When we examine the results for inference time (Figure 7.8) and model size (Figure 7.9), we observe that the difference for the Pamap2-DeepConvLSTM model is much more pronounced compared to other dataset-architecture pairs when using pruning methods versus the *Lite* model. In pruning methods, the efficiency gain from pruning increases with the size of the initial model (in our case, *OM*). A similar trend is seen in [164], where a VGG-16 model with 138 million parameters experienced a 13-fold reduction, whereas a SqueezeNet model with 1.2 million parameters only saw a 3.2-fold reduction. Therefore, in our analysis, the most significant advantage is observed for the Pamap2-DeepConvLSTM model, which has the highest number of parameters (see the *Params* column of Table 4.4). For the remaining dataset-architecture pairs, the *Lite* models and the pruning models required almost the same inference time. In terms of model size, except for the Sensors-DeepConvLSTM pair, the models have 1.0-1.9x smaller sizes compared to *Lite* for *CP* and 1.0-1.7x smaller sizes compared to *Lite* for *PDP*.

The model size results (Figure 7.7) show an increase in performance for models using SqueezeNet with both *CP* and *PDP* methods. This increase is 2.30% for Opportunity and 2.38% for Sensors with *CP*, while it is slightly lower with *PDP* (0.93% for Opportunity and 1.98% for Sensors). For DeepConvLSTM, there is only a slight performance decrease for Wisdm with *CP* compared to *OM* (0.06%). In all other dataset-architecture pairs, there is a performance increase. For all four datasets, models that show more performance improvement with *CP* also show more improvement

with *PDP*. Similar improvements are observed. When using DeepConvLSTM, the performance improvements with *CP* are highest for Opportunity, followed by Sensors, Pamap2, and Wisdm. The order is the same for *PDP*. This indicates that a model performing well with one technique is likely to perform well with another technique.

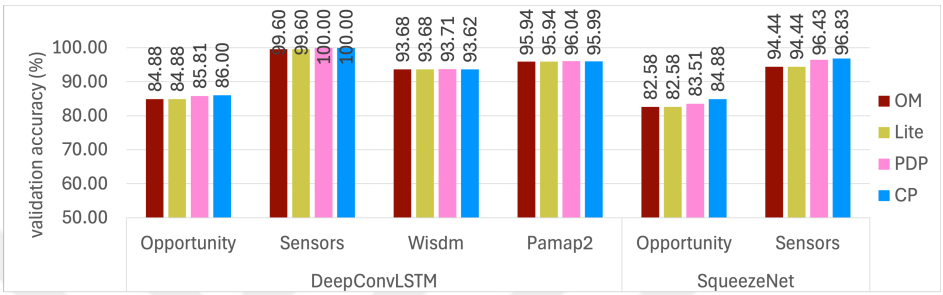


Figure 7.7. Validation accuracy (%) of pruned models with pruning rate 60%.

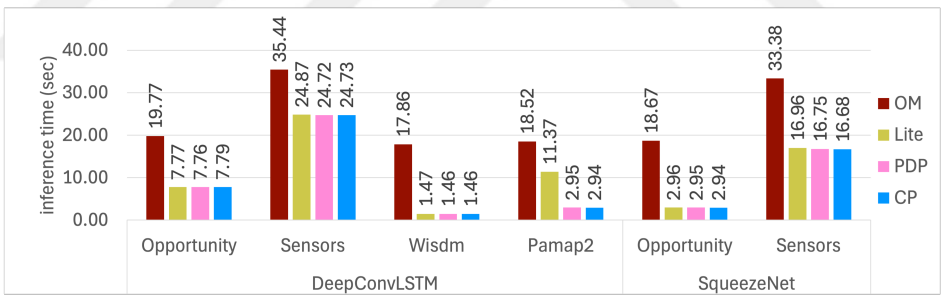


Figure 7.8. Inference time (sec) of pruned models with pruning rate 60%.

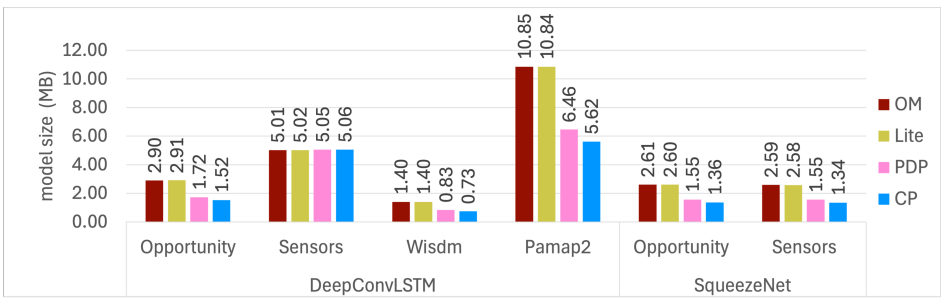


Figure 7.9. Model size (MB) of pruned models with pruning rate 60%.

In summary, pruning techniques such as *CP* and *PDP* are effective methods for enhancing model efficiency by reducing computational and memory requirements while preserving performance. *CP* provides a straightforward and consistent pruning approach, which makes it simple and manageable to use. With *PDP*, the pruning schedule can be more fine-tuned, potentially resulting in better performance. According to our analysis, models with larger initial sizes, like Pamap2-DeepConvLSTM, benefit more from pruning. There are, however, some datasets that are more robust to high pruning rates than others, depending on their characteristics. Therefore, the selection of a pruning strategy should be based on the specific needs and constraints of the deployment environment, taking into account the trade-offs between model size reduction and performance accuracy.

7.4. Recognition Success and Resource Consumption Comparison of Compressed Original Models Versus Boosted LM

In this section, we compare three versions of the models *OM*, *MM*, and *LM* for DeepConvLSTM and SqueezeNet across various datasets. We also evaluate the performance of *LM* models boosted with attention and knowledge distillation techniques and compare these with the *Lite*, quantized, and pruned versions of *OM* models to address RQ10. The results are presented in Figure 7.10, 7.11 and 7.12. Note that validation accuracy results for Opportunity-SqueezeNet are not included in Figure 7.12. The reason for this is that the performance results drop to as low as 22%, which necessitates a wider range on the x-axis. This makes it harder to visually observe the changes in validation accuracy for other dataset-architecture pairs. Therefore, we excluded these results from the figure and provided only in text form. The validation accuracy results for Opportunity-SqueezeNet are as follows: 82.58%, 68.51%, 22.28%, 58.68%, 22.28%, 22.28%, 59.80%, 82.58%, 82.76%, 93.68%, 84.88% and 83.51% for *OM*, *MM*, *LM*, *LM-Att*, *RB-KD*, *RAB-KD*, *RB-KD-Att*, *Lite*, *DRQ*, *FQ*, *CP*, and *PDP*, respectively.

Reducing the model size from *OM* to *LM* has a significant impact on model size and recognition performance, though it only slightly affects inference time. The model

size reduction between *OM* (1.40-10.85MB) and *LM* (10KB-57KB) ranges from 80.5x to 195.6x. Similarly, the model size of *MM* (28KB-195KB) is 32.9x to 57.1x less than *OM*. The minimal impact on inference time might be due to the powerful CPU used during the experiments (Apple M3 Pro chip with 11-core CPU). Testing on devices with limited computational power, such as an Arduino Nano, could provide further insights into inference time performance under constrained conditions.

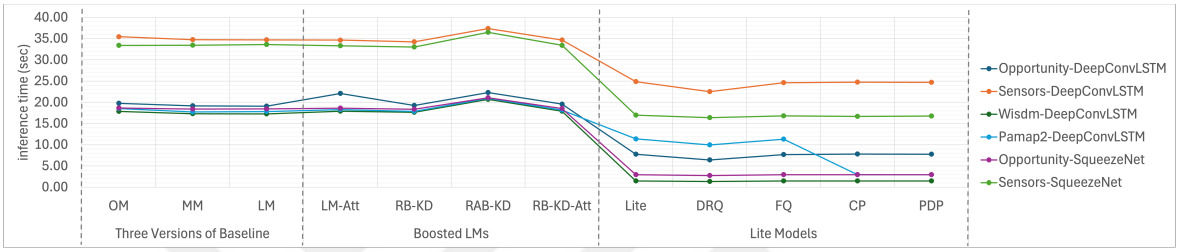


Figure 7.10. Inference time (sec) comparison of original models, boosted LM models and compressed original models.

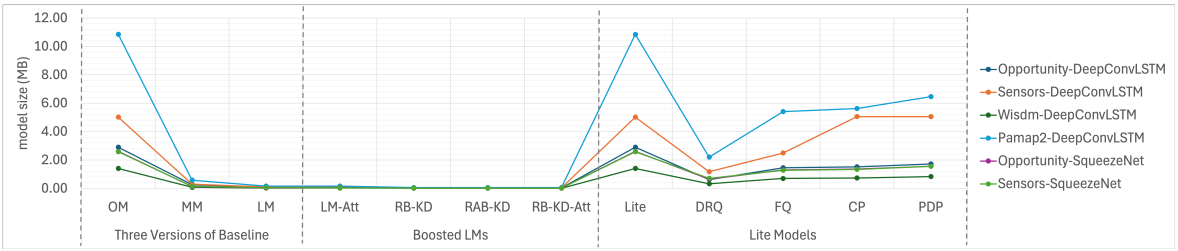


Figure 7.11. Model size (MB) comparison of original models, boosted LM models and compressed original models.

For *LM* models, the application of attention and distillation techniques (individually as *LM-Att* and *RB-KD*, and in combination as *RAB-KD* and *RB-KD-Att*) resulted in a slight increase in inference time, particularly when both techniques are used together. However, this did not significantly increase the model size. *KD* alone does not increase model size, while integrating attention into distillation provides enhanced performance without a notable increase in size, with a maximum increase of 1.5KB.

Furthermore, these models require less than 256KB of storage, enabling deployment on a wide range of MCU/IoT devices, such as those listed in Table 1.1.

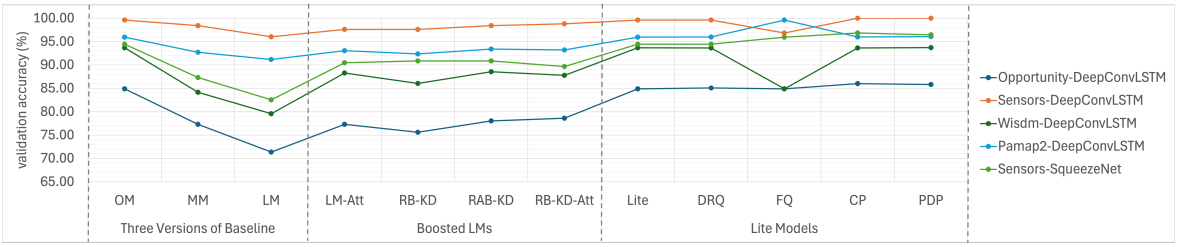


Figure 7.12. Validation accuracy (%) comparison of original models, boosted LM models and compressed original models.

While the *OM* model maintains its recognition performance, the *Lite* models, particularly those using the *DRQ* method, show significant advantages in both compression and recognition success. The *DRQ* method reduces the model size by up to 5x, whereas other techniques (*FQ*, *CP* and *PDP*) achieve reductions of up to 2x. However, even with the most compressed model (using *DRQ* on the Pamap2-DeepConvLSTM), which decreases the model size from 10.85MB to 2.20MB, the resulting size may still be too large for devices with memory limitations, such as Arduino Zero with 256KB of Flash memory. On the other hand, boosted *LM* models outperform the recognition performance of *MM*, except for the Opportunity-SqueezeNet model, with a significantly smaller model size. For example, considering Wisdm-DeepConvLSTM, the initial validation accuracy of *OM*, *MM* and *LM* is 93.68%, 84.16% and 79.56%, respectively. The boosted *LM*'s accuracy obtained with *RAB-KD* is 88.56%, which not only surpasses *MM*'s performance but also falls nearly between *OM* and *MM*. Besides, the boosted *LM* (11KB) requires 130x less size than *OM* (1.40MB) and 2.5x less size than *MM* (28KB). Therefore, the boosted *LM* models are the most memory-efficient, offering performance that, while not as high as *OM*, generally falls between *OM* and *MM*. This level of recognition success may be sufficient for non-critical HAR application domains [165] like fitness and well-being. Here, there is a trade-off between the recognition success of the model, the computational power of the device, and the device's memory.

7.5. Conclusion and Insights

This chapter presented a comprehensive analysis of the performance of boosted *LM*s compared with compressed models created using quantization and pruning techniques. The primary aim was to explore the trade-offs between model size, recognition performance, computational efficiency, and memory usage.

The reduction in model size from *OM*s to *LM*s was substantial, with decreases ranging from 80.5x to 195.6x. *MM* also exhibited significant size reductions, ranging from 32.9x to 57.1x. These reductions are crucial for deploying models on devices with limited memory capacity. Despite the considerable reduction in model size, the inference time was minimally affected, likely due to the powerful CPU used in the experiments (Apple M3 Pro chip). Future testing on less powerful devices like the Arduino Nano could provide further insights into inference time under more constrained conditions.

The application of attention and knowledge distillation techniques to *LM* models (individually as *LM-Att* and *RB-KD*, and in combination as *RAB-KD* and *RB-KD-Att*) led to performance improvements with only a slight increase in inference time. Knowledge distillation alone did not increase model size, while the combination of attention and distillation provided enhanced performance without a notable increase in size. When compared with compressed models, the boosted *LM* models approached the performance of *MM* models but did not match the recognition success of *OM* models. Quantized and pruned versions of *OM* models showed varied success, with quantization generally more efficient.

This chapter highlighted the importance of balancing recognition performance with resource efficiency. Techniques such as quantization and pruning effectively reduced computational and memory demands, making models more suitable for deployment on resource-constrained devices. The optimization logic provided by TensorFlow Lite played a significant role in reducing model size and resource consumption.

A clear trade-off exists between recognition success, computational power, and device memory. For devices with severe memory limitations such as Nordic Thingy:52 [6] (with 32KB SRAM and 256KB Flash memory) and Arduino Due [10] (with 96KB SRAM and 512KB Flash memory), lightweight models with attention mechanisms (*LM-Att*) or response-based knowledge distillation (*RB-KD*) are recommended. Quantization should be prioritized for efficiency, with pruning serving as a supplementary method. Further examples of these memory-constrained devices can be found in Chapter 1.

In summary, the integration of attention mechanisms and knowledge distillation appears to offer a promising approach to enhancing the performance of lightweight HAR models. These techniques facilitate the development of high-performing, resource-efficient models suitable for deployment on tiny devices, addressing the challenges posed by resource-constrained environments.

8. CONCLUSION

In this thesis, we investigated the improvement of lightweight sensor-based HAR models to enhance their applicability on resource-constrained devices. To this end, we addressed the key research question: *"How can we achieve a better balance between recognition accuracy and resource efficiency by enhancing lightweight sensor-based HAR models instead of relying on larger ones?"*. To answer, we explored the use of the attention mechanism and KD individually and in combination. The results showed that the performance of the boosted lightweight models is leveraged by both attention and KD techniques. More specifically, the boosted lightweight models achieved performance comparable to a moderate-size model while using 2.6x-3.7x fewer resources, demonstrating its potential for real-world deployment. Besides we also compared these methods with two well-known compression techniques, quantization and pruning, to evaluate their effectiveness. Our comprehensive analysis addressed ten specific research questions, providing detailed insights into the performance and efficiency of the proposed models.

8.1. Findings

The findings of this thesis are organized according to the research questions addressed in each chapter. Below, we summarize the key results and insights obtained from our investigations.

First, in Chapter 4, we addressed three research questions concerning the integration of attention mechanisms into deep learning models for sensor-based HAR. RQ1 examined the impact of incorporating channel and spatial attention modules individually on the recognition performance of the DeepConvLSTM and SqueezeNet architectures. Our findings indicated that spatial attention generally outperformed channel attention across various datasets, with improvements in recognition accuracy and f1-scores, particularly for the Opportunity and Sensors datasets. RQ2 focused on evaluating the

effects of varying parameters, such as reduction ratios for channel attention and kernel sizes for spatial attention. The results showed that optimal parameter settings could significantly enhance model performance while maintaining efficient resource consumption. Lastly, RQ3 explored the combined application of channel and spatial attention modules. The combined approach yielded the highest recognition performance improvements, highlighting the synergy between channel and spatial attention mechanisms. These findings underscore the effectiveness of attention modules in enhancing the performance of HAR models, with a negligible increase in the number of parameters. Then, in Chapter 5, we examined the effect of incorporating attention modules into different sizes of DeepConvLSTM and SqueezeNet models on recognition accuracy and resource consumption under RQ4. Our results showed that attention mechanisms significantly enhanced the performance of lightweight models, allowing them to match or exceed the accuracy of larger models while remaining resource-efficient.

Then, in Chapter 6, we investigated the potential of using attention mechanisms and KD to enhance the performance of lightweight models. We addressed four research questions. RQ5 explored the impact of individually using attention and response-based distillation techniques on the recognition performance of lightweight models. The findings revealed that attention mechanisms consistently outperformed KD regarding recognition performance. RQ6 examined the extent to which incorporating attention mechanisms into KD, specifically through attention-based distillation, could further boost the performance of lightweight models. Our findings indicated that attention-based distillation outperformed traditional response-based distillation, providing substantial gains in recognition accuracy while maintaining efficient resource usage. Finally, RQ7 explored a KD system where the teacher model does not include attention modules. This approach proved effective, showing that the student models with attention mechanisms could achieve superior performance, even when the teacher models lacked these mechanisms.

Finally, in Chapter 7, we addressed three research questions concerning the comparison of boosted lightweight models with compressed models using quantization and

pruning techniques. RQ8 examined the effectiveness of different quantization methods in reducing model size while maintaining recognition performance. The quantization analysis, which investigated FQ and DRQ, showed that these methods significantly reduced model size, with DRQ achieving the most substantial reductions while maintaining recognition performance. RQ9 focused on the impact of pruning techniques. The analysis demonstrated that applying pruning techniques, CP and PDP, significantly reduced model size and improved inference time while maintaining high recognition accuracy by removing redundant weights. Lastly, RQ10 involved a comparative analysis of the recognition success and resource consumption between compressed original and boosted lightweight models. The findings revealed that while boosted lightweight models with attention and distillation techniques approached the performance of moderate-sized models, they did not entirely match the recognition success of original models. However, they offered a compelling trade-off between recognition accuracy, computational efficiency, and memory usage, making them suitable for deployment on resource-constrained devices.

8.2. Insights Gained and Recommendations

In this section, we discuss several strategies and recommendations for implementing sensor-based HAR inference on resource-constrained devices. Based on our findings, different approaches can be considered depending on the availability of pre-trained models, memory constraints, and hardware support. Figure 8.1 presents a flowchart that outlines the recommended approach for creating lightweight models, considering these key factors.

Training lightweight models with attention mechanisms is a viable approach for scenarios where a pre-trained large model is not available. This involves training a lightweight model tailored to the device's resource constraints and adding attention mechanisms to enhance its performance without significantly increasing resource consumption. This allows the model to achieve performance levels comparable to larger models by effectively focusing on the most informative parts of the features.

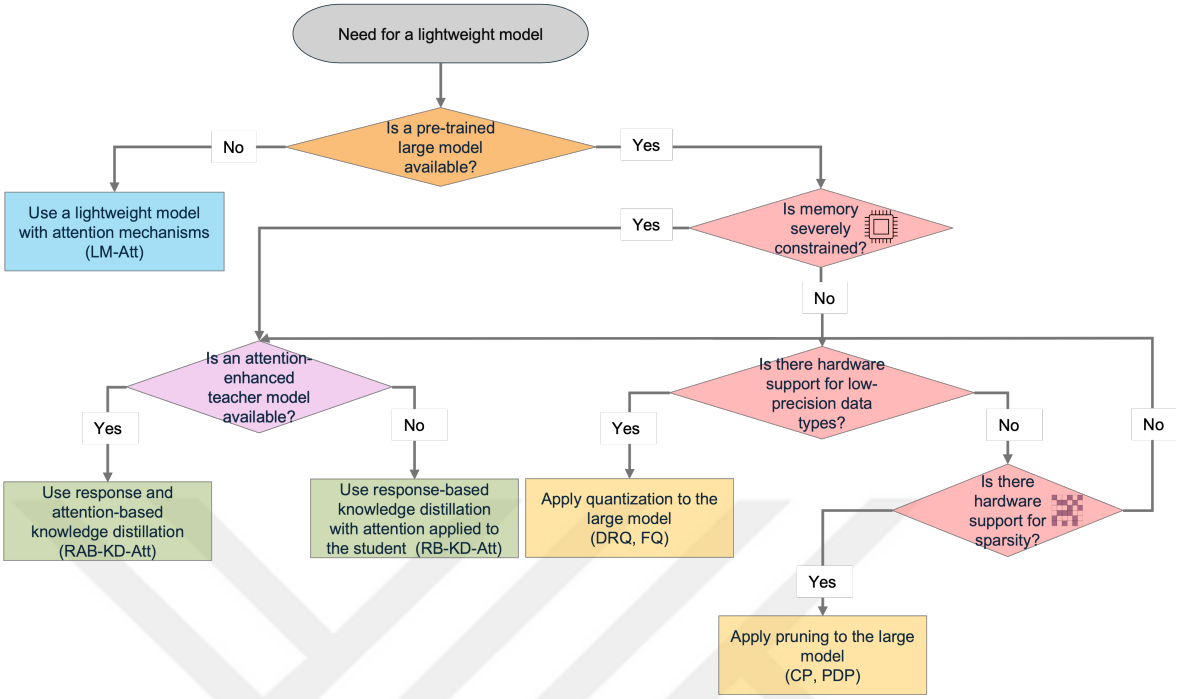


Figure 8.1. Flowchart for lightweight model selection based on insights gained and recommendations.

If a pre-trained large model is available, we can design a lightweight model and improve its performance using response-based knowledge distillation. This method involves using the larger model as a teacher to transfer knowledge to the lightweight model, enhancing its performance compared to a lightweight model trained from scratch. Additionally, if an attention-enhanced teacher model is available, or if sufficient data is available to train the teacher model with attention, the response and attention-based knowledge distillation approach can be considered. This method leverages both the predictions and attention maps from the teacher to significantly boost the performance of the lightweight student model.

In cases where memory constraints are not severe, and the target device supports TensorFlow Lite, quantization and pruning techniques can be employed. Quantization generally outperforms pruning in terms of efficiency and compatibility with standard hardware, reducing the precision of the model parameters to achieve a smaller model size and lower inference time without significantly impacting recognition performance.

Pruning, on the other hand, eliminates certain parameters within the model, making it sparse and potentially incompatible with standard hardware. Additionally, pruning typically requires extra training to fine-tune the pruned model. Therefore, quantization should be prioritized when efficiency is critical, and pruning can be considered as a supplementary method.

It is crucial to consider the hardware limitations and capabilities of the target deployment environment, as not all operations are supported by every device and also by TensorFlow Lite. TensorFlow Lite's optimization logic focuses on reducing the size and resource consumption of the model, making it essential to test the optimized model on the target device to ensure its suitability.

Our analysis indicates that there is a trade-off between model size, recognition success, computational power, and memory availability. Boosted lightweight models, while achieving performance levels close to or even higher than moderate models, achieve considerable improvements but remain below the performance of the original models. Therefore, it is important to balance these factors based on the specific needs and constraints of the application. For applications on devices with very limited memory, the required model size may still be too large, necessitating further optimization or simplification of the model.

In conclusion, the choice of model design and optimization techniques should be tailored to the specific constraints and requirements of the deployment environment. By leveraging lightweight models, attention mechanisms, knowledge distillation, and appropriate hardware-specific optimizations, efficient and high-performing sensor-based HAR systems can be achieved even for resource-constrained devices.

8.3. Limitations and Future Directions

This study has certain limitations. The compression techniques used in this study, quantization and pruning, may not always fully achieve the theoretical efficiency gains

and sometimes fail to provide consistent results. This inconsistency highlights the difficulty of balancing model size reduction and maintaining recognition performance. Additionally, the experiments were conducted on a high-performance Apple M3 Pro chip, which might not accurately reflect the performance on more constrained devices like Arduino Nano or other low-power microcontrollers. The significant difference in computational power can lead to varying inference times and performance outcomes. Furthermore, many existing studies often rely on a single dataset, limiting the robustness and adaptability of their models. Although this study evaluated models across four diverse datasets, further testing on more varied and larger datasets is necessary to enhance generalizability.

Despite these limitations, there are promising future directions for this research. To ensure the practical applicability of the models, extensive testing on a wide range of hardware with varying computational power and memory capacity is essential. This includes low-power microcontrollers, various wearables, and other edge devices. Additionally, incorporating on-device training capabilities for microcontrollers would enable models to adapt and improve based on new data collected on the device, addressing the issue of decreasing accuracy over time without real-time updates.

Increasing the number and diversity of datasets used for training and validation can enhance the generalizability of the models, ensuring they are robust and adaptable to different environments and activities. Exploring the combination of various optimization techniques, such as applying compression to the boosted lightweight models, is another promising direction. Integrating methods like attention mechanisms with model compression techniques can potentially yield highly efficient models without compromising performance. This hybrid approach is worth investigating further to find the optimal balance between resource usage and recognition accuracy.

Furthermore, keeping the HAR model in the cloud where privacy is not a concern is also a viable option. Cloud-based solutions require edge devices to transfer data, and a comparative analysis can be performed between the data transfer time for the large

model in the cloud and the time elapsed for the lightweight model on-device. In addition to time, it is also important to measure the energy consumption associated with wireless data transmission. This analysis can provide insights into the efficiency and effectiveness of HAR resources, offering a broader perspective on the best deployment strategies for different use cases.

REFERENCES

1. Lattanzi, E., M. Donati and V. Freschi, "Exploring Artificial Neural Networks Efficiency in Tiny Wearable Devices for Human Activity Recognition", *MDPI Sensors*, Vol. 22, No. 7, pp. 2637–2658, 2022.

2. Khan, D., N. Al Mudawi, M. Abdelhaq, A. Alazeb, S. S. Alotaibi, A. Algarni and A. Jalal, "A Wearable Inertial Sensor Approach for Locomotion and Localization Recognition on Physical Activity", *MDPI Sensors*, Vol. 24, No. 3, pp. 735–763, 2024.

3. He, Z., Y. Sun and Z. Zhang, "Human Activity Recognition Based on Deep Learning Regardless of Sensor Orientation", *Applied Sciences*, Vol. 14, No. 9, pp. 3637–3658, 2024.

4. Bian, S., M. Liu, B. Zhou and P. Lukowicz, "The State-of-the-Art Sensing Techniques in Human Activity Recognition: A Survey", *MDPI Sensors*, Vol. 22, No. 12, pp. 4596–4625, 2022.

5. Chen, K., D. Zhang, L. Yao, B. Guo, Z. Yu and Y. Liu, "Deep Learning for Sensor-Based Human Activity Recognition: Overview, Challenges, and Opportunities", *Computing Surveys*, Vol. 54, No. 4, pp. 1–40, 2021.

6. "Nordic Thingy:52 Prototyping Platform - nordicsemi.com", `https://www.nordicsemi.com/Products/Development-hardware/Nordic-Thingy-52`, accessed on July 30, 2024.

7. "STM32F407VG – High-Performance Foundation Line, Arm Cortex-M4 Core With DSP and FPU, 1 Mbyte of Flash Memory, 168 MHz CPU, ART Accelerator, Ethernet, FSMC – STMicroelectronics", `https://www.st.com/en/microcontrollers-microprocessors/stm32f407vg.`

html#documentation, accessed on July 30, 2024.

8. "STM32F746NG – High-Performance and DSP With FPU, Arm Cortex-M7 MCU With 1 Mbyte of Flash Memory, 216 MHz CPU, Art Accelerator, L1 Cache, SDRAM, TFT – STMicroelectronics", https://www.st.com/en/microcontrollers-microprocessors/stm32f746ng.html, accessed on July 30, 2024.

9. "Arduino Zero — Arduino Online Shop", https://store-usa.arduino.cc/products/arduino-zero, accessed on July 30, 2024.

10. "Arduino Due — Arduino Online Shop", https://store-usa.arduino.cc/products/arduino-due, accessed on July 30, 2024.

11. "Arduino Nano 33 BLE — Arduino Online Shop", https://store-usa.arduino.cc/products/arduino-nano-33-ble, accessed on July 30, 2024.

12. "pico-w-datasheet.pdf", https://datasheets.raspberrypi.com/picow/pico-w-datasheet.pdf, accessed on July 30, 2024.

13. Diab, M. S. and E. Rodriguez-Villegas, "Embedded Machine Learning Using Microcontrollers in Wearable and Ambulatory Systems for Health and Care Applications: A Review", *IEEE Access*, Vol. 10, pp. 98450–98474, 2022.

14. Prakash, S., M. Stewart, C. Banbury, M. Mazumder, P. Warden, B. Plancher and V. J. Reddi, "Is TinyML Sustainable?", *Communications of the ACM*, Vol. 66, No. 11, pp. 68–77, 2023.

15. Reddi, V. J., B. Plancher, S. Kennedy, L. Moroney, P. Warden, A. Agarwal, C. R. Banbury, M. Banzi, M. Bennett, B. Brown, S. Chitlangia, R. Ghosal, S. Grafman, R. Jaeger, S. Krishnan, M. Lam, D. Leiker, C. Mann, M. Mazumder, D. Pajak, D. Ramaprasad, J. E. Smith, M. Stewart and D. Tingley, "Widening Access to

Applied Machine Learning With TinyML", *arXiv:2106.04008*, 2021.

16. "Hexiwear - MIKROE", `https://www.mikroe.com/hexiwear`, accessed on July 30, 2024.

17. Cesari, P., M. Cristani, F. Demrozi, F. Pascucci, P. M. Picotti, G. Pravadelli, C. Tomazzoli, C. Turetta, T. C. Workneh and L. Zenti, "Towards Posture and Gait Evaluation through Wearable-Based Biofeedback Technologies", *MDPI Electronics*, Vol. 12, No. 3, pp. 644–655, 2023.

18. Zhang, S., Y. Li, S. Zhang, F. Shahabi, S. Xia, Y. Deng and N. Alshurafa, "Deep Learning in Human Activity Recognition With Wearable Sensors: A Review on Advances", *MDPI Sensors*, Vol. 22, No. 4, pp. 1476–1519, 2022.

19. Alawneh, L., T. Alsarhan, M. Al-Zinati, M. Al-Ayyoub, Y. Jararweh and H. Lu, "Enhancing Human Activity Recognition Using Deep Learning and Time Series Augmented Data", *Journal of Ambient Intelligence and Humanized Computing*, Vol. 12, pp. 1–16, 2021.

20. Liu, R., A. A. Ramli, H. Zhang, E. Henricson and X. Liu, "An Overview of Human Activity Recognition Using Wearable Sensors: Healthcare and Artificial Intelligence", *Conference on Internet of Things*, p. 1, Virtual Event, 2021.

21. Xu, X., E. Nemati, K. Vatanparvar, V. Nathan, T. Ahmed, M. M. Rahman, D. McCaffrey, J. Kuang and J. A. Gao, "Listen2cough: Leveraging End-to-End Deep Learning Cough Detection Model To Enhance Lung Health Assessment Using Passively Sensed Audio", *Interactive, Mobile, Wearable and Ubiquitous Technologies*, Vol. 5, No. 1, pp. 1–22, 2021.

22. Bhattacharya, D., D. Sharma, W. Kim, M. F. Ijaz and P. K. Singh, "Ensem-HAR: An Ensemble Deep Learning Model for Smartphone Sensor-Based Human Activity Recognition for Measurement of Elderly Health Monitoring", *MDPI Biosensors*,

Vol. 12, No. 6, pp. 393–418, 2022.

23. Steels, T., B. Van Herbruggen, J. Fontaine, T. De Pessemier, D. Plets and E. De Poorter, "Badminton Activity Recognition Using Accelerometer Data", *MDPI Sensors*, Vol. 20, No. 17, pp. 4685–4701, 2020.

24. Javed, A. R., M. U. Sarwar, S. Khan, C. Iwendi, M. Mittal and N. Kumar, "Analyzing the Effectiveness and Contribution of Each Axis of Tri-Axial Accelerometer Sensor for Accurate Activity Recognition", *MDPI Sensors*, Vol. 20, No. 8, pp. 2216–2234, 2020.

25. Zhou, Q., Z. Qu, S. Guo, B. Luo, J. Guo, Z. Xu and R. Akerkar, "On-Device Learning Systems for Edge Intelligence: A Software and Hardware Synergy Perspective", *IEEE Internet of Things*, Vol. 8, No. 15, pp. 11916–11934, 2021.

26. Blalock, D., J. J. Gonzalez Ortiz, J. Frankle and J. Guttag, "What Is the State of Neural Network Pruning?", *Proceedings of Machine Learning and Systems*, Vol. 2, No. 1, pp. 129–146, 2020.

27. Xu, S., A. Huang, L. Chen and B. Zhang, "Convolutional Neural Network Pruning: A Survey", *39th Chinese Control Conference*, p. 7458, Shenyang, China, 2020.

28. Liang, T., J. Glossner, L. Wang, S. Shi and X. Zhang, "Pruning and Quantization for Deep Neural Network Acceleration: A Survey", *Neurocomputing*, Vol. 461, pp. 370–403, 2021.

29. Jacob, B., S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam and D. Kalenichenko, "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference", *Conference on Computer Vision and Pattern Recognition*, p. 2704, Salt Lake City, United States of America, 2018.

30. Krishnamoorthi, R., "Quantizing Deep Convolutional Networks for Efficient In-

ference: A Whitepaper", *arXiv:1806.08342*, 2018.

31. Guo, Y., "A Survey on Methods and Theories of Quantized Neural Networks", *arXiv:1808.04752*, 2018.

32. Hohman, F., M. B. Kery, D. Ren and D. Moritz, "Model Compression in Practice: Lessons Learned from Practitioners Creating On-device Machine Learning Experiences", *Conference on Human Factors in Computing Systems*, p. 1, Honolulu, United States of America, 2024.

33. "TensorFlow Lite — ML for Mobile and Edge Devices", `https://www.tensorflow.org/lite/`, accessed on July 07, 2024.

34. Wen, Y. and D. Gregg, "Exploiting Weight Redundancy in CNNs: Beyond Pruning and Quantization", *arXiv:2006.11967*, 2020.

35. Contoli, C. and E. Lattanzi, "A Study on the Application of TensorFlow Compression Techniques to Human Activity Recognition", *IEEE Access*, Vol. 11, pp. 48046–48058, 2023.

36. Reyes-Ortiz, J.-L., L. Oneto, A. Samà, X. Parra and D. Anguita, "Transition-Aware Human Activity Recognition Using Smartphones", *Neurocomputing*, Vol. 171, pp. 754–767, 2016.

37. Daghero, F., A. Burrello, C. Xie, M. Castellano, L. Gandolfi, A. Calimera, E. Macii, M. Poncino and D. J. Pagliari, "Human Activity Recognition on Microcontrollers With Quantized and Adaptive Deep Neural Networks", *Transactions on Embedded Computing Systems*, Vol. 21, No. 4, pp. 1–28, 2022.

38. Ghibellini, A., L. Bononi and M. Di Felice, "Intelligence at the IoT Edge: Activity Recognition With Low-Power Microcontrollers and Convolutional Neural Networks", *19th Annual Consumer Communications and Networking Conference*, p. 701, Las Vegas, United States of America, 2022.

39. Bursa, S. Ö., Ö. Durmaz İncel and G. Işıklar Alptekin, "Building Lightweight Deep Learning Models With TensorFlow Lite for Human Activity Recognition on Mobile Devices", *Annals of Telecommunications*, Vol. 78, No. 11, pp. 687–702, 2023.

40. Zebin, T., P. J. Scully, N. Peek, A. J. Casson and K. B. Ozanyan, "Design and Implementation of a Convolutional Neural Network on an Edge Computing Smartphone for Human Activity Recognition", *IEEE Access*, Vol. 7, pp. 133509–133520, 2019.

41. Ordóñez, F. J. and D. Roggen, "Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition", *MDPI Sensors*, Vol. 16, No. 1, pp. 115–140, 2016.

42. Iandola, F. N., S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally and K. Keutzer, "SqueezeNet: AlexNet-Level Accuracy With 50x Fewer Parameters and <0.5 MB Model Size", *arXiv:1602.07360*, 2016.

43. Hinton, G., O. Vinyals and J. Dean, "Distilling the Knowledge in a Neural Network", *arXiv:1503.02531*, 2015.

44. Gou, J., B. Yu, S. J. Maybank and D. Tao, "Knowledge Distillation: A Survey", *International Journal of Computer Vision*, Vol. 129, No. 6, pp. 1789–1819, 2021.

45. Chen, R., H. Luo, F. Zhao, X. Meng, Z. Xie and Y. Zhu, "Modeling Accurate Human Activity Recognition for Embedded Devices Using Multi-level Distillation", *arXiv:2107.07331*, 2021.

46. Deng, S., J. Chen, D. Teng, C. Yang, D. Chen, T. Jia and H. Wang, "LHAR: Lightweight Human Activity Recognition on Knowledge Distillation", *Journal of Biomedical and Health Informatics*, Vol. 28, No. 10, pp. 1–10, 2023.

47. Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez,

Ł. Kaiser and I. Polosukhin, "Attention Is All You Need", *Advances in Neural Information Processing Systems*, Vol. 30, pp. 5998–6008, 2017.

48. Zagoruyko, S. and N. Komodakis, "Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer", *arXiv:1612.03928*, 2016.

49. Zeng, M., H. Gao, T. Yu, O. J. Mengshoel, H. Langseth, I. Lane and X. Liu, "Understanding and Improving Recurrent Networks for Human Activity Recognition by Continuous Attention", *Symposium on Wearable Computers*, p. 56, Singapore, Singapore, 2018.

50. Gao, W., L. Zhang, Q. Teng, J. He and H. Wu, "DanHAR: Dual Attention Network for Multimodal Human Activity Recognition Using Wearable Sensors", *Applied Soft Computing*, Vol. 111, pp. 107728–107740, 2021.

51. Yao, S., Y. Zhao, H. Shao, D. Liu, S. Liu, Y. Hao, A. Piao, S. Hu, S. Lu and T. F. Abdelzaher, "Sadeepsense: Self-Attention Deep Learning Framework for Heterogeneous On-Device Sensors in Internet of Things Applications", *Conference on Computer Communications*, p. 1243, Paris, France, 2019.

52. Woo, S., J. Park, J.-Y. Lee and I. S. Kweon, "CBAM: Convolutional Block Attention Module", *European Conference on Computer Vision*, p. 3, Munich, Germany, 2018.

53. Yang, Y., H. Wang, R. Jiang, X. Guo, J. Cheng and Y. Chen, "A Review of IoT-enabled Mobile Healthcare: Technologies, Challenges, and Future Trends", *IEEE Internet of Things*, Vol. 9, No. 12, pp. 9478–9502, 2022.

54. Hassan, M. M., M. Z. Uddin, A. Mohamed and A. Almogren, "A Robust Human Activity Recognition System Using Smartphone Sensors and Deep Learning", *Future Generation Computer Systems*, Vol. 81, pp. 307–313, 2018.

55. Agac, S., M. Shoaib and O. Durmaz Incel, "Smoking Recognition With Smartwatch Sensors in Different Postures and Impact of User's Height", *Journal of Ambient Intelligence and Smart Environments*, Vol. 12, No. 3, pp. 239–261, 2020.

56. Kyritsis, K., C. Diou and A. Delopoulos, "Food Intake Detection From Inertial Sensors Using LSTM Networks", *Conference on Image Analysis and Processing*, p. 411, Catania, Italy, 2017.

57. Khan, N. S. and M. S. Ghani, "A Survey of Deep Learning Based Models for Human Activity Recognition", *Wireless Personal Communications*, Vol. 120, No. 2, pp. 1593–1635, 2021.

58. Barshan, B. and A. Yurtman, "Classifying Daily and Sports Activities Invariantly to the Positioning of Wearable Motion Sensor Units", *IEEE Internet of Things*, Vol. 7, No. 6, pp. 4801–4815, 2020.

59. Yurtman, A., B. Barshan and S. Redif, "Position Invariance for Wearables: Interchangeability and Single-Unit Usage via Machine Learning", *IEEE Internet of Things*, Vol. 8, No. 10, pp. 8328–8342, 2020.

60. Demrozi, F., G. Pravadelli, A. Bihorac and P. Rashidi, "Human Activity Recognition Using Inertial, Physiological and Environmental Sensors: A Comprehensive Survey", *IEEE Access*, Vol. 8, pp. 210816–210836, 2020.

61. Beddiar, D. R., B. Nini, M. Sabokrou and A. Hadid, "Vision-Based Human Activity Recognition: A Survey", *Multimedia Tools and Applications*, Vol. 79, No. 41, pp. 30509–30555, 2020.

62. Ahmad, I., A. Ullah and W. Choi, "WiFi-Based Human Sensing With Deep Learning: Recent Advances, Challenges, and Opportunities", *Open Journal of the Communications Society*, Vol. 5, pp. 3595–3623, 2024.

63. Hussain, A., K. Zafar, A. R. Baig, R. Almakki, L. AlSuwaidan and S. Khan,

"Sensor-Based Gym Physical Exercise Recognition: Data Acquisition and Experiments", *MDPI Sensors*, Vol. 22, No. 7, pp. 2489–2506, 2022.

64. Vavoulas, G., C. Chatzaki, T. Malliotakis, M. Pediaditis and M. Tsiknakis, "The Mobiact Dataset: Recognition of Activities of Daily Living Using Smartphones", *International Conference on Information and Communication Technologies for Ageing Well and e-Health*, Vol. 2, p. 143, Rome, Italy, 2016.

65. Shoaib, M., S. Bosch, O. D. Incel, H. Scholten and P. J. Havinga, "Fusion of Smartphone Motion Sensors for Physical Activity Recognition", *MDPI Sensors*, Vol. 14, No. 6, pp. 10146–10176, 2014.

66. Huan, R., C. Jiang, L. Ge, J. Shu, Z. Zhan, P. Chen, K. Chi and R. Liang, "Human Complex Activity Recognition With Sensor Data Using Multiple Features", *IEEE Sensors*, Vol. 22, No. 1, pp. 757–775, 2021.

67. Gravina, R., P. Alinia, H. Ghasemzadeh and G. Fortino, "Multi-Sensor Fusion in Body Sensor Networks: State-of-the-Art and Research Challenges", *Information Fusion*, Vol. 35, pp. 68–80, 2017.

68. Jiang, W. and Z. Yin, "Human Activity Recognition Using Wearable Sensors by Deep Convolutional Neural Networks", *23rd Conference on Multimedia*, p. 1307, Brisbane, Australia, 2015.

69. Chen, Y. and Y. Xue, "A Deep Learning Approach to Human Activity Recognition Based on Single Accelerometer", *Conference on Systems, Man, and Cybernetics*, p. 1488, Hong Kong, China, 2015.

70. Grzeszick, R., J. M. Lenk, F. M. Rueda, G. A. Fink, S. Feldhorst and M. ten Hompel, "Deep Neural Network Based Human Activity Recognition for the Order Picking Process", *4th Workshop on Sensor-Based Activity Recognition and Interaction*, p. 1, Rostock, Germany, 2017.

71. Abedin, A., M. Ehsanpour, Q. Shi, H. Rezatofighi and D. C. Ranasinghe, "Attend and Discriminate: Beyond the State-of-the-Art for Human Activity Recognition Using Wearable Sensors", *Interactive, Mobile, Wearable and Ubiquitous Technologies*, Vol. 5, No. 1, pp. 1–22, 2021.

72. Huynh-The, T., C.-H. Hua, N. A. Tu and D.-S. Kim, "Physical Activity Recognition With Statistical-Deep Fusion Model Using Multiple Sensory Data for Smart Health", *IEEE Internet of Things*, Vol. 8, No. 3, pp. 1533–1543, 2020.

73. Hanif, M. A., T. Akram, A. Shahzad, M. A. Khan, U. Tariq, J.-I. Choi, Y. Nam and Z. Zulfiqar, "Smart Devices Based Multisensory Approach for Complex Human Activity Recognition", *Computers, Materials and Continua*, Vol. 70, pp. 3221–3234, 2022.

74. Pires, I. M., N. Pombo, N. M. Garcia and F. Flórez-Revuelta, "Multi-Sensor Mobile Platform for the Recognition of Activities of Daily Living and Their Environments Based on Artificial Neural Networks", *Conference on Artificial Intelligence*, p. 5850, Stockholm, Sweden, 2018.

75. Sena, J., J. Barreto, C. Caetano, G. Cramer and W. R. Schwartz, "Human Activity Recognition Based on Smartphone and Wearable Sensors Using Multiscale DCNN Ensemble", *Neurocomputing*, Vol. 444, pp. 226–243, 2021.

76. Brauwers, G. and F. Frasincar, "A General Survey on Attention Mechanisms in Deep Learning", *Transactions on Knowledge and Data Engineering*, Vol. 35, No. 4, pp. 3279–3298, 2021.

77. Liu, S., S. Yao, J. Li, D. Liu, T. Wang, H. Shao and T. Abdelzaher, "Globalfusion: A Global Attentional Deep Learning Framework for Multisensor Information Fusion", *Interactive, Mobile, Wearable and Ubiquitous Technologies*, Vol. 4, No. 1, pp. 1–27, 2020.

78. Li, Y., L. Wang and F. Liu, "Multi-Branch Attention-Based Grouped Convolution Network for Human Activity Recognition Using Inertial Sensors", *MDPI Electronics*, Vol. 11, No. 16, pp. 2526–2540, 2022.

79. Akter, M., S. Ansary, M. A.-M. Khan and D. Kim, "Human Activity Recognition Using Attention-Mechanism-Based Deep Learning Feature Combination", *MDPI Sensors*, Vol. 23, No. 12, pp. 5715–5730, 2023.

80. Reiss, A. and D. Stricker, "Introducing a New Benchmarked Dataset for Activity Monitoring", *16th Symposium on Wearable Computers*, p. 108, Newcastle, United Kingdom, 2012.

81. "Daphnet Freezing of Gait - UCI Machine Learning Repository", `https://archive.ics.uci.edu/dataset/245/daphnet+freezing+of+gait`, accessed on August 12, 2024.

82. Zappi, P., C. Lombriser, T. Stiefmeier, E. Farella, D. Roggen, L. Benini and G. Tröster, "Activity Recognition From On-Body Sensors: Accuracy-Power Trade-Off by Dynamic Sensor Selection", *Wireless Sensor Networks: 5th European Conference*, p. 17, Bologna, Italy, 2008.

83. Ma, H., W. Li, X. Zhang, S. Gao and S. Lu, "AttnSense: Multi-level Attention Mechanism For Multimodal Human Activity Recognition", *28th Joint Conference on Artificial Intelligence*, p. 3109, Macao, China, 2019.

84. Stisen, A., H. Blunck, S. Bhattacharya, T. S. Prentow, M. B. Kjærgaard, A. Dey, T. Sonne and M. M. Jensen, "Smart Devices Are Different: Assessing and Mitigatingmobile Sensing Heterogeneities for Activity Recognition", *13th Conference on Embedded Networked Sensor Systems*, p. 127, Seoul, South Korea, 2015.

85. Kwapisz, J. R., G. M. Weiss and S. A. Moore, "Activity Recognition Using Cell Phone Accelerometers", *Special Interest Group on Knowledge Discovery and Data*

Mining Explorations Newsletter, Vol. 12, No. 2, pp. 74–82, 2011.

86. Chavarriaga, R., H. Sagha, A. Calatroni, S. T. Digumarti, G. Tröster, J. d. R. Millán and D. Roggen, "The Opportunity Challenge: A Benchmark Database for On-Body Sensor-Based Activity Recognition", *Pattern Recognition Letters*, Vol. 34, No. 15, pp. 2033–2042, 2013.

87. Micucci, D., M. Mobilio and P. Napoletano, "Unimib Shar: A Dataset for Human Activity Recognition Using Acceleration Data From Smartphones", *Applied Sciences*, Vol. 7, No. 10, pp. 1101–1120, 2017.

88. Shoaib, M., S. Bosch, O. D. Incel, H. Scholten and P. J. Havinga, "Complex Human Activity Recognition Using Smartphone and Wrist-Worn Motion Sensors", *MDPI Sensors*, Vol. 16, No. 4, pp. 426–450, 2016.

89. Tang, Y., L. Zhang, Q. Teng, F. Min and A. Song, "Triple Cross-Domain Attention on Human Activity Recognition Using Wearable Sensors", *Transactions on Emerging Topics in Computational Intelligence*, Vol. 6, No. 5, pp. 1167–1176, 2022.

90. Anguita, D., A. Ghio, L. Oneto, X. Parra and J. L. Reyes-Ortiz, "A Public Domain Dataset for Human Activity Recognition Using Smartphones", *The European Symposium on Artificial Neural Networks*, Vol. 3, p. 3, Bruges, Belgium, 2013.

91. Murahari, V. S. and T. Plötz, "On Attention Models for Human Activity Recognition", *Symposium on Wearable Computers*, p. 100, Singapore, Singapore, 2018.

92. Tao, W., H. Chen, M. Moniruzzaman, M. C. Leu, Z. Yi and R. Qin, "Attention-Based Sensor Fusion for Human Activity Recognition Using IMU Signals", *arXiv:2112.11224*, 2021.

93. Wang, Y., H. Xu, Y. Liu, M. Wang, Y. Wang, Y. Yang, S. Zhou, J. Zeng, J. Xu,

S. Li and J. Li, "A Novel Deep Multifeature Extraction Framework Based on Attention Mechanism Using Wearable Sensor Data for Human Activity Recognition", *IEEE Sensors*, Vol. 23, No. 7, pp. 7188–7198, 2023.

94. Tan, T.-H., Y.-L. Chang, J.-R. Wu, Y.-F. Chen and M. Alkhaleefah, "Convolutional Neural Network with Multi-Head Attention for Human Activity Recognition", *IEEE Internet of Things*, Vol. 11, No. 2, pp. 3032–3043, 2023.

95. Li, Z., H. Li and L. Meng, "Model Compression for Deep Neural Networks: A Survey", *MDPI Computers*, Vol. 12, No. 3, pp. 60–82, 2023.

96. Howard, A. G., M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto and H. Adam, "Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications", *arXiv:1704.04861*, 2017.

97. Zhang, X., X. Zhou, M. Lin and J. Sun, "Shufflenet: An Extremely Efficient Convolutional Neural Network for Mobile Devices", *Conference on Computer Vision and Pattern Recognition*, p. 6848, Salt Lake City, United States of America, 2018.

98. Cai, H., J. Lin, Y. Lin, Z. Liu, H. Tang, H. Wang, L. Zhu and S. Han, "Enable Deep Learning on Mobile Devices: Methods, Systems, and Applications", *ACM Transactions on Design Automation of Electronic Systems*, Vol. 27, No. 3, pp. 1–50, 2022.

99. Véstias, M. P., R. P. Duarte, J. T. de Sousa and H. C. Neto, "Moving Deep Learning to the Edge", *MDPI Algorithms*, Vol. 13, No. 5, pp. 125–158, 2020.

100. Shuvo, M. M. H., S. K. Islam, J. Cheng and B. I. Morshed, "Efficient Acceleration of Deep Learning Inference on Resource-Constrained Edge Devices: A Review", *Proceedings of the IEEE*, Vol. 111, No. 1, pp. 42–91, 2022.

101. Krizhevsky, A., I. Sutskever and G. E. Hinton, "Imagenet Classification With Deep Convolutional Neural Networks", *Advances in Neural Information Process-*

ing Systems, Vol. 25, pp. 1097–1105, 2012.

102. Zoph, B. and Q. V. Le, "Neural Architecture Search With Reinforcement Learning", *arXiv:1611.01578*, 2016.

103. Yamada, F., S. Tsuji, H. Kawaguchi, A. Inoue and Y. Sakai, "A High-Speed Neural Architecture Search Considering the Number of Weights", *Advances in Artificial Intelligence*, p. 109, Virtual Event, 2021.

104. Chen, W., X. Gong and Z. Wang, "Neural Architecture Search on Imagenet in Four GPU Hours: A Theoretically Inspired Perspective", *arXiv:2102.11535*, 2021.

105. Ometov, A., V. Shubina, L. Klus, J. Skibinska, S. Saafi, P. Pascacio, L. Flueratoru, D. Quezada-Gaibor, N. Chukhno, O. Chukhno, A. Ali, A. Channa, E. Svertoka, W. B. Qaim, R. C. Marqués, S. Holcer, J. Torres-Sospedra, S. Casteleyn, G. Ruggeri, G. Araniti, R. Burget, J. Hosek and E. S. Lohan, "A Survey on Wearable Technology: History, State-of-the-Art and Current Challenges", *Computer Networks*, Vol. 193, pp. 108074–108111, 2021.

106. Dian, F. J., R. Vahidnia and A. Rahmati, "Wearables and the Internet of Things (IoT), Applications, Opportunities, and Challenges: A Survey", *IEEE Access*, Vol. 8, pp. 69200–69211, 2020.

107. Tmamna, J., E. B. Ayed and M. B. Ayed, "Deep Learning for Internet of Things in Fog Computing: Survey and Open Issues", *5th Conference on Advanced Technologies for Signal and Image Processing*, p. 1, Sousse, Tunisia, 2020.

108. Wang, J., Y. Chen, S. Hao, X. Peng and L. Hu, "Deep Learning for Sensor-Based Activity Recognition: A Survey", *Pattern Recognition Letters*, Vol. 119, pp. 3–11, 2019.

109. Ahad, M. A. R., A. D. Antar, M. Ahmed, M. A. R. Ahad, A. D. Antar and M. Ahmed, "Sensor-Based Human Activity Recognition: Challenges Ahead", *IoT*

Sensor-Based Activity Recognition: Human Activity Recognition, Vol. 173, pp. 175–189, 2021.

110. Tang, Y., Q. Teng, L. Zhang, F. Min and J. He, "Layer-Wise Training Convolutional Neural Networks With Smaller Filters for Human Activity Recognition Using Wearable Sensors", *IEEE Sensors*, Vol. 21, No. 1, pp. 581–592, 2020.

111. Bock, M., A. Hölzemann, M. Moeller and K. Van Laerhoven, "Improving Deep Learning for HAR with Shallow LSTMs", *Symposium on Wearable Computers*, p. 7, Virtual Event, 2021.

112. Xu, C., D. Chai, J. He, X. Zhang and S. Duan, "InnoHAR: A Deep Neural Network for Complex Human Activity Recognition", *IEEE Access*, Vol. 7, pp. 9893–9902, 2019.

113. Rad, N. M., A. Bizzego, S. M. Kia, G. Jurman, P. Venuti and C. Furlanello, "Convolutional Neural Network for Stereotypical Motor Movement Detection in Autism", *arXiv:1511.01865*, 2015.

114. Yang, J., M. N. Nguyen, P. P. San, X. L. Li and S. Krishnaswamy, "Deep Convolutional Neural Networks on Multichannel Time Series for Human Activity Recognition", *24th Joint Conference on Artificial Intelligence*, p. 3995, Buenos Aires, Argentina, 2015.

115. Yao, S., S. Hu, Y. Zhao, A. Zhang and T. Abdelzaher, "Deepsense: A Unified Deep Learning Framework for Time-Series Mobile Sensing Data Processing", *26th Conference on World Wide Web*, p. 351, Perth, Australia, 2017.

116. Peppas, K., A. C. Tsolakis, S. Krinidis and D. Tzovaras, "Real-Time Physical Activity Recognition on Smart Mobile Devices Using Convolutional Neural Networks", *Applied Sciences*, Vol. 10, No. 23, pp. 8482–8507, 2020.

117. Ignatov, A., "Real-Time Physical Activity Recognition on Smart Mobile Devices

Using Convolutional Neural Networks", *Applied Soft Computing*, Vol. 62, pp. 915–922, 2018.

118. Lockhart, J. W., G. M. Weiss, J. C. Xue, S. T. Gallagher, A. B. Grosner and T. T. Pulickal, "Design Considerations for the WISDM Smart Phone-Based Sensor Mining Architecture", *5th Workshop on Knowledge Discovery from Sensor Data*, p. 25, San Diego, California, 2011.

119. Gkountelos, D., M. Kokhazadeh, C. Bournas, G. Keramidas and V. Kelefouras, "Towards Highly Compressed CNN Models for Human Activity Recognition in Wearable Devices", *Signal Processing: Algorithms, Architectures, Arrangements, and Applications*, p. 83, Poznan, Poland, 2023.

120. Zhou, Y., H. Zhao, Y. Huang, T. Riedel, M. Hefenbrock and M. Beigl, "TinyHAR: A Lightweight Deep Learning Model Designed for Human Activity Recognition", *Symposium on Wearable Computers*, p. 89, Cambridge, United Kingdom, 2022.

121. Barshan, B. and M. C. Yüksek, "Recognizing Daily and Sports Activities in Two Open Source Machine Learning Environments Using Body-Worn Sensor Units", *The Computer*, Vol. 57, No. 11, pp. 1649–1667, 2014.

122. Zou, H., Z. Chen, C. Zhang, A. Yuan, B. Wang, L. Wang, J. Li and Y. Pan, "STFNet: Enhanced and Lightweight Spatiotemporal Fusion Network for Wearable Human Activity Recognition", *IEEE Sensors*, Vol. 24, No. 8, pp. 13686–13698, 2024.

123. Baños, O., M. Damas, H. Pomares, I. Rojas, M. A. Tóth and O. Amft, "A Benchmark Dataset To Evaluate Sensor Displacement in Activity Recognition", *Conference on Ubiquitous Computing*, p. 1026, Pittsburgh, Pennsylvania, 2012.

124. Zhang, M. and A. A. Sawchuk, "USC-HAD: A Daily Activity Dataset for Ubiquitous Activity Recognition Using Wearable Sensors", *Conference on Ubiquitous*

Computing, p. 1036, Pittsburgh, Pennsylvania, 2012.

125. Ankita, S. Rani, H. Babbar, S. Coleman, A. Singh and H. M. Aljahdali, "An Efficient and Lightweight Deep Learning Model for Human Activity Recognition Using Smartphones", *MDPI Sensors*, Vol. 21, No. 11, pp. 3845–3862, 2021.

126. Coelho, Y. L., F. d. A. S. dos Santos, A. Frizera-Neto and T. F. Bastos-Filho, "A Lightweight Framework for Human Activity Recognition on Wearable Devices", *IEEE Sensors*, Vol. 21, No. 21, pp. 24471–24481, 2021.

127. Oresti, B., G. Rafael, A. Juan, D. Miguel, P. Hector, R. Ignacio, S. Alejandro and V. Claudia, "mHealthDroid: A Novel Framework for Agile Development of Mobile Health Applications", *Ambient Assisted Living and Daily Activities*, Vol. 8868, pp. 91–98, 2014.

128. Mardanpour, M., M. Sepahvand, F. Abdali-Mohammadi, M. Nikouei and H. Sarabi, "Human Activity Recognition Based on Multiple Inertial Sensors Through Feature-Based Knowledge Distillation Paradigm", *Information Sciences*, Vol. 640, pp. 119073–119086, 2023.

129. Ni, J., A. H. Ngu and Y. Yan, "Progressive Cross-Modal Knowledge Distillation for Human Action Recognition", *30th Conference on Multimedia*, p. 5903, Lisbon, Portugal, 2022.

130. Yuan, Z., Z. Yang, H. Ning and X. Tang, "Multiscale Knowledge Distillation With Attention Based Fusion for Robust Human Activity Recognition", *Scientific Reports*, Vol. 14, No. 1, pp. 12411–12427, 2024.

131. Ni, J., R. Sarbajna, Y. Liu, A. H. Ngu and Y. Yan, "Cross-Modal Knowledge Distillation for Vision-to-Sensor Action Recognition", *Conference on Acoustics, Speech and Signal Processing*, p. 4448, Singapore, Singapore, 2022.

132. Jeon, E. S., A. Som, A. Shukla, K. Hasanaj, M. P. Buman and P. Turaga, "Role

of Data Augmentation Strategies in Knowledge Distillation for Wearable Sensor Data", *IEEE Internet of Things*, Vol. 9, No. 14, pp. 12848–12860, 2021.

133. Ullah, H. and A. Munir, "A 3dcnn-Based Knowledge Distillation Framework for Human Activity Recognition", *Journal of Imaging*, Vol. 9, No. 4, pp. 82–109, 2023.

134. Qu, Y., W. Deng and J. Hu, "H-AT: Hybrid Attention Transfer for Knowledge Distillation", *Pattern Recognition and Computer Vision: 3rd Chinese Conference*, p. 249, Nanjing, China, 2020.

135. Cho, J. H. and B. Hariharan, "On the Efficacy of Knowledge Distillation", *Conference on Computer Vision*, p. 4794, Seoul, Korea, 2019.

136. Zhou, Z., C. Zhuge, X. Guan and W. Liu, "Channel Distillation: Channel-Wise Attention for Knowledge Distillation", *arXiv:2006.01683*, 2020.

137. Krizhevsky, A., *Learning Multiple Layers of Features from Tiny Images*, Tech. rep., University of Toronto, 2009.

138. Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, "Imagenet: A Large-Scale Hierarchical Image Database", *Conference on Computer Vision and Pattern Recognition*, p. 248, Miami, United States of America, 2009.

139. He, K., X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition", *Conference on Computer Vision and Pattern Recognition*, p. 770, Las Vegas, United States of America, 2016.

140. Bursa, S. O., O. D. Incel and G. I. Alptekin, "Transforming Deep Learning Models for Resource-Efficient Activity Recognition on Mobile Devices", *5th Conference on Cloud and Internet of Things*, p. 83, Marrakech, Morocco, 2022.

141. Contoli, C. and E. Lattanzi, "Energy Efficiency of Deep Learning Compression

Techniques in Wearable Human Activity Recognition", *Conference on Artificial Intelligence Applications and Innovations*, Vol. 675, p. 102, Leon, Spain, 2023.

142. Lin, J., L. Zhu, W.-M. Chen, W.-C. Wang and S. Han, "Tiny Machine Learning: Progress and Futures", *Circuits and Systems Magazine*, Vol. 23, No. 3, pp. 8–34, 2023.

143. Siirtola, P., H. Koskimäki and J. Röning, "Openhar: A Matlab Toolbox for Easy Access To Publicly Open Human Activity Data Sets", *International Joint Conference and International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*, p. 1396, Singapore, Singapore, 2018.

144. Abadi, M., A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. J. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Józefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. G. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. A. Tucker, V. Vanhoucke, V. Vasudevan, F. B. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu and X. Zheng, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems", *arXiv:1603.04467*, 2016.

145. Jouini, O., K. Sethom, A. Namoun, N. Aljohani, M. H. Alanazi and M. N. Alanazi, "A Survey of Machine Learning in Edge Computing: Techniques, Frameworks, Applications, Issues, and Research Directions", *MDPI Technologies*, Vol. 12, No. 6, pp. 81–115, 2024.

146. "TensorFlow Lite", `https://www.tensorflow.org/lite/guide`, accessed on June 24, 2024.

147. Zhang, X., Y. Wang, S. Lu, L. Liu, L. Xu and W. Shi, "OpenEI: An Open Framework for Edge Intelligence", *39th Conference on Distributed Computing Systems*, p. 1840, Dallas, United States of America, 2019.

148. "Core ML — Apple Developer Documentation", https://developer.apple.com/documentation/coreml/, accessed on August 12, 2024.

149. "Caffe2 — A New Lightweight, Modular, and Scalable Deep Learning Framework", https://caffe2.ai/, accessed on August 12, 2024.

150. Chen, T., M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang and Z. Zhang, "Mxnet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems", *arXiv:1512.01274*, 2015.

151. "TensorFlow Lite and TensorFlow Operator Compatibility", https://www.tensorflow.org/lite/guide/ops_compatibility, accessed on June 24, 2024.

152. "On-Device Training with TensorFlow Lite", https://www.tensorflow.org/lite/examples/on_device_training/overview#retrain_the_model_on_a_device, accessed on June 24, 2024.

153. "TensorFlow Lite for Microcontrollers", https://www.tensorflow.org/lite/microcontrollers, accessed on June 24, 2024.

154. Garcia-Perez, A., R. Miñón, A. I. Torre-Bastida and E. Zulueta-Guerrero, "Analysing Edge Computing Devices for the Deployment of Embedded AI", *MDPI Sensors*, Vol. 23, No. 23, pp. 9495–9516, 2023.

155. Singh, S. P., M. K. Sharma, A. Lay-Ekuakille, D. Gangwar and S. Gupta, "Deep ConvLSTM With Self-Attention for Human Activity Decoding Using Wearable Sensors", *IEEE Sensors*, Vol. 21, No. 6, pp. 8575–8582, 2020.

156. Gupta, S., "Deep Learning Based Human Activity Recognition (HAR) Using Wearable Sensor Data", *International Journal of Information Management Data Insights*, Vol. 1, No. 2, pp. 100046–100064, 2021.

157. Guo, P., Z. Li and Y. Sun, "Research on Action Pattern Recognition Based on DeepConvLSTM", *6th Conference on Information Systems and Computer Aided Education*, p. 1100, Dalian, China, 2023.

158. Xia, K., J. Huang and H. Wang, "LSTM-CNN Architecture for Human Activity Recognition", *IEEE Access*, Vol. 8, pp. 56855–56866, 2020.

159. Lv, T., X. Wang, L. Jin, Y. Xiao and M. Song, "A Hybrid Network Based on Dense Connection and Weighted Feature Aggregation for Human Activity Recognition", *IEEE Access*, Vol. 8, pp. 68320–68332, 2020.

160. Guo, M.-H., T.-X. Xu, J.-J. Liu, Z.-N. Liu, P.-T. Jiang, T.-J. Mu, S.-H. Zhang, R. R. Martin, M.-M. Cheng and S.-M. Hu, "Attention Mechanisms in Computer Vision: A Survey", *Computational Visual Media*, Vol. 8, No. 3, pp. 331–368, 2022.

161. Romero, A., N. Ballas, S. E. Kahou, A. Chassang, C. Gatta and Y. Bengio, "Fitnets: Hints for Thin Deep Nets", *arXiv:1412.6550*, 2014.

162. Kuzmin, A., M. Nagel, M. Van Baalen, A. Behboodi and T. Blankevoort, "Pruning vs Quantization: Which Is Better?", *Advances in Neural Information Processing Systems*, Vol. 36, pp. 62414–62427, 2024.

163. Guerra, L. and T. Drummond, "Automatic Pruning for Quantized Neural Networks", *Digital Image Computing: Techniques and Applications*, p. 1, Gold Coast, Australia, 2021.

164. Han, S., *Efficient Methods and Hardware for Deep Learning*, Ph.D. Thesis, Stanford University, 2017.

165. Manokhin, M., P. Chollet and P. Desgreys, "Towards Flexible and Low-Power Wireless Smart Sensors: Reconfigurable Analog-to-Feature Conversion for Healthcare Applications", *MDPI Sensors*, Vol. 24, No. 3, pp. 999–1027, 2024.

166. Menghani, G., "Efficient Deep Learning: A Survey on Making Deep Learning Models Smaller, Faster, and Better", *ACM Computing Surveys*, Vol. 55, No. 12, pp. 1–37, 2023.

167. Kumar, P. and S. Suresh, "mHAR: A Novel Convolutional Recurrent Model for Recognizing Motion-Based Human Activity", *Procedia Computer Science*, Vol. 229, pp. 218–227, 2023.

168. Chen, R., H. Luo, F. Zhao, X. Meng, Z. Xie and Y. Zhu, "Modeling Accurate Human Activity Recognition for Embedded Devices Using Multi-Level Distillation", *arXiv:2107.07331*, 2021.

169. Xu, Q., M. Wu, X. Li, K. Mao and Z. Chen, "Contrastive Distillation With Regularized Knowledge for Deep Model Compression on Sensor-Based Human Activity Recognition", *Transactions on Industrial Cyber-Physical Systems*, Vol. 1, pp. 217–226, 2023.

170. Gholami, A., S. Kim, Z. Dong, Z. Yao, M. W. Mahoney and K. Keutzer, "A Survey of Quantization Methods for Efficient Neural Network Inference", *arXiv:2103.13630*, 2021.

171. Agac, S. and O. D. Incel, "User Authentication and Identification on Smart Glasses with Motion Sensors", *Springer Nature Computer Science*, Vol. 4, No. 6, pp. 761–775, 2023.

172. Yang, C., X. Yu, Z. An and Y. Xu, "Categories of Response-Based, Feature-Based, and Relation-Based Knowledge Distillation", *arXiv:2306.10687*, 2023.

173. Aldeer, M., D. Waterworth, Z. Hussain, T. Chowdhury, C. Brito, Q. Z. Sheng, R. P. Martin and J. Ortiz, "Medbuds: In-Ear Inertial Medication Taking Detection Using Smart Wireless Earbuds", *2nd Workshop on Cyber-Physical-Human System Design and Implementation*, p. 19, Milan, Italy, 2022.