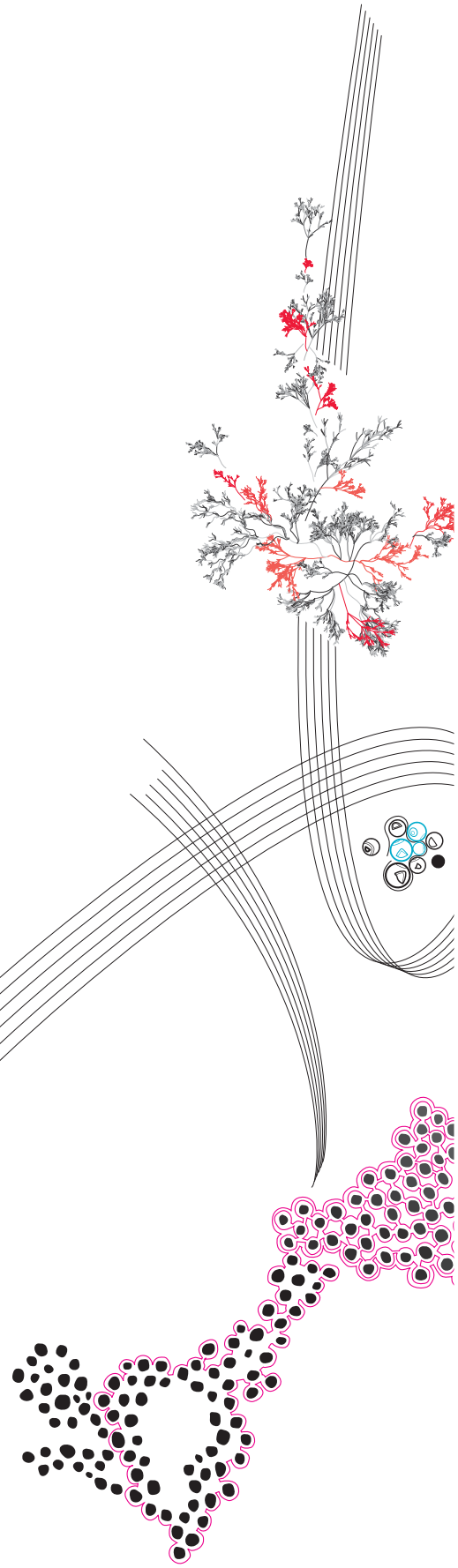

MSc Embedded Systems
Final Project

RESOURCE-EFFICIENT DEEP LEARNING FOR MOBILE ACTIVITY RECOGNITION ON EDGE DEVICES

Yixiang Lu

Supervisor:

Asst. Prof. Yanqiu Huang
Assoc. Prof. Özlem Durmaz Incel

January, 2025

Department of Embedded Systems
Faculty of Electrical Engineering,
Mathematics and Computer Science,
University of Twente

UNIVERSITY OF TWENTE.

Contents

Abstract

This thesis investigates methods to improve the efficiency and performance of deep-learning for sensor-based Human Activity Recognition on limited-resource edge devices, using the DeepConvLSTM as a case study. The primary objective is to reduce model size and inference time while maintaining high accuracy. To achieve this, existing techniques such as quantization, pruning, knowledge distillation, and attention mechanisms were evaluated for their effectiveness and compatibility when combined. The optimized models, integrating these techniques, were successfully deployed on the Arduino Nano 33 BLE Sense. This deployment demonstrates the feasibility of running DeepConvLSTM models enhanced with knowledge distillation and attention mechanisms on tiny TensorFlow Lite for Microcontrollers edge devices.

To further improve efficiency in sensor-based Human Activity Recognition, this thesis introduced event-driven inference, inspired by Neural Architecture Search and Dynamic Routing Networks. The event-driven model dynamically selects the most suitable path for each input sample, significantly improving efficiency while maintaining high accuracy. By combining event-driven inference with existing optimization techniques, the proposed models significantly reduce inference time while achieving slightly higher accuracy than the original models. They only exhibit 7.85% to 35% of the inference time comparing with the original model, and the size is also largely reduced. These results underscore the effectiveness of event-driven inference in Human Activity Recognition tasks, showcasing its potential to greatly enhance efficiency and performance on resource-constrained edge devices.

Chapter 1

Introduction

With the growing demand for healthcare and sports monitoring, Human Activity Recognition (HAR) using machine learning has seen significant development. Concerns about privacy, the need for real-time processing, and the rapid advancements in microcontroller units (MCU) have made the deployment of sensor-based machine learning models on edge devices an increasingly attractive solution for HAR [11]. The deployment of machine learning models, particularly deep architectures, on edge devices requires careful consideration of multiple factors. First, the model must be small enough to fit within the limited memory of the edge device. Additionally, the model must be computationally efficient to function effectively on an MCU with limited processing power and energy constraints. Compatibility issues must also be addressed to ensure smooth deployment and operation.

This thesis explores efficient techniques for the deployment of deep learning models on edge devices, focusing on optimizing model size and inference efficiency, specifically for sensor-based HAR tasks. The primary aim is to develop a solution that ensures the model remains compact and computationally efficient while maintaining satisfactory accuracy, as well as compatibility with the resource-constrained environments of edge devices.

The models are trained on four distinct sensor-based datasets: Opportunity [5], Sensors [20], Wisdm [17], Pamap2 [18]. Each dataset is characterized by different numbers of activity classes, sampling rates, noise levels, class imbalances, and feature sets. These datasets incorporate data from a variety of sensors, including accelerometers, gyroscopes, and magnetometers, to capture a wide range of motion patterns, and they are commonly used in the literature for benchmarking studies. The DeepConvLSTM [16] architecture has been selected as the primary model due to its proven effectiveness in time-series classification tasks. Additionally, other models, such as LightGBM, are explored as alternative architectures for benchmarking and comparative analysis.

There are several ways to make machine learning models smaller and more efficient, such as using quantization [3, 7, 10] and pruning techniques. At the same time, methods like knowledge distillation [8, 13] and attention mechanism [2, 25] can enhance the performance of small models. These approaches are especially useful for maintaining accuracy even when the models are downsized to work on devices with limited resources, such as edge devices. Moreover, there are techniques like Neural Architecture Search and Dynamic Routing Networks, which search for the optimal path to find the balance between accuracy and efficiency.

Quantization has been extensively studied and proven to be an effective technique for improving the efficiency of machine learning models [3, 7, 10]. However, there is limited discussion in these literature regarding the compatibility of full integer quantization with TensorFlow Lite for Microcontrollers (TFLM) and the potential accuracy degradation it

may cause, often due to issues like overfitting, leaving it as the ***first research gap*** to be solved. In this thesis, the underlying causes of accuracy loss are investigated and addressed. Specifically, the problem is mitigated by reducing model complexity through approaches such as feature selection in Section 3.3, ensuring that the optimized models achieve both efficiency and robust performance.

Knowledge distillation is a well-established technique in which a smaller student model learns from a larger, more accurate teacher model, effectively improving the performance of the student model with minimal computational cost [8, 13]. Similarly, attention mechanisms have been shown to significantly improve model performance by enabling the model to focus on the most critical layers or features [25]. However, many studies have overlooked the trade-offs in model size and inference time, which are critical for deploying models on resource-constrained platforms. This thesis will thoroughly test these aspects and address this ***second research gap***. Moreover, the integration of knowledge distillation, attention mechanisms, and quantization has not been extensively explored. Additionally, the combination of these methods has rarely been tested on edge devices, leaving issues of compatibility unresolved, such as encountered in [6]. This is marked as the ***third research gap*** in this thesis.

Neural Architecture Search (NAS) and Dynamic Routing Networks (DRN) are advanced methods used to improve model efficiency by identifying the best inference path [26, 22, 4]. However, these methods come with challenges. NAS requires significant computational resources and time during the training phase, as each potential architecture must be tested and evaluated [24]. DRN, on the other hand, adds extra computational overhead and increases storage requirements due to the dynamic path selection process [14]. To address this ***fourth research gap***, building on these concepts, we propose implementing **Event-driven Inference** for applications on edge devices. This approach aims to further reduce inference time by finding the suitable inference path while maintaining a minimal cost in terms of model size and training effort.

To address these gaps, this thesis thoroughly evaluates the effects of quantization, pruning, knowledge distillation, and attention mechanisms on model accuracy, size, and inference time in Chapter 3. The ***first research gap*** is addressed in Section 3.3 and the ***second research gap*** is addressed in Section 3.5 and 3.6. Based on the results from Chapter 3, Chapter 4 proposes a streamlined approach to create an extremely compact model that maintains high performance with only 3% of the original size. Some layers of the model are redesigned to ensure compatibility with TFLM. The proposed model is rigorously tested for overall performance and compatibility with TFLM, validating the feasibility of combining these optimization techniques. Furthermore, the study demonstrates the compatibility of attention mechanisms with TFLM and evaluates the interplay of these methods when used together. After that, the optimized model, based on the DeepConvLSTM architecture, is successfully deployed on the Arduino Nano 33 BLE Sense in Section 4.3. This achievement underscores the practicality of applying attention mechanisms and other optimization techniques to edge devices, paving the way for efficient and high-performing models in resource-constrained environments and answering the ***third research gap***.

Event-driven Inference, inspired by Neural Architecture Search (NAS) and Dynamic Routing Networks (DRN), is designed to improve model efficiency by predefining the inference path for different types of input samples. The approach involves pre-classifying the input samples based on their primary characteristics. In this thesis, the samples are categorized into two groups: simple and complex, or momentary and durative. Following this initial classification, specialized expert networks are employed to accurately predict the

final label for each sample, optimizing performance and efficiency. This approach reduces the overhead of dynamically selecting paths during inference and eliminates the need for complex path searching during training, making it a more practical and efficient solution for the ***fourth research gap***. In Chapter 5, we deployed and tested the event-driven models on a Raspberry Pi 5. The results demonstrated significant reductions in inference time up to 70% with minimal impact on model size. To address the challenges that arise from the deployment of multiple collaborative models, we redesigned the architecture using a DeepConvLSTM framework to integrate the event-driven model into a unified system in Chapter 6. This integration simplifies deployment and ensures compatibility for a wide range of applications on resource-constrained devices, as well as further reduce the inference time up to 86.7% and even increase the accuracy.

We have formulated specific research questions to guide the objectives and discoveries of this thesis. These RQs are designed to clarify the approach and findings while focusing on practical applications for machine learning techniques on resource-constrained devices. Figure 1.1 illustrates the organization of this thesis, making the location of the answers to the RQs.

- RQ1: What is the impact of individual quantization, pruning, knowledge distillation and attention mechanism on the performance and efficiency of the DeepConvLSTM model across different HAR datasets?

  This is answered in Chapter 3

- RQ2: Is an attention mechanism feasible and suitable for a model deployed on edge devices with TFLM?

  This is answered in Section 4.3

- RQ3: How can we improve performance by combining these individually tested techniques?

  This is answered in Section 4.2 and 4.3

- RQ4: Is there a streamlined process to obtain a compact model with minimal adverse effects on performance, while keeping the model compatible with TFLM?

  This is answered in Section 4.1

- RQ5: What effect does an event-driven inference have on the performance, inference time, and size of the model with different HAR datasets?

  This is answered in Chapter 5

- RQ6: Is this event-driven model compatible with devices running TFLM?

  This is answered in Section 6.2

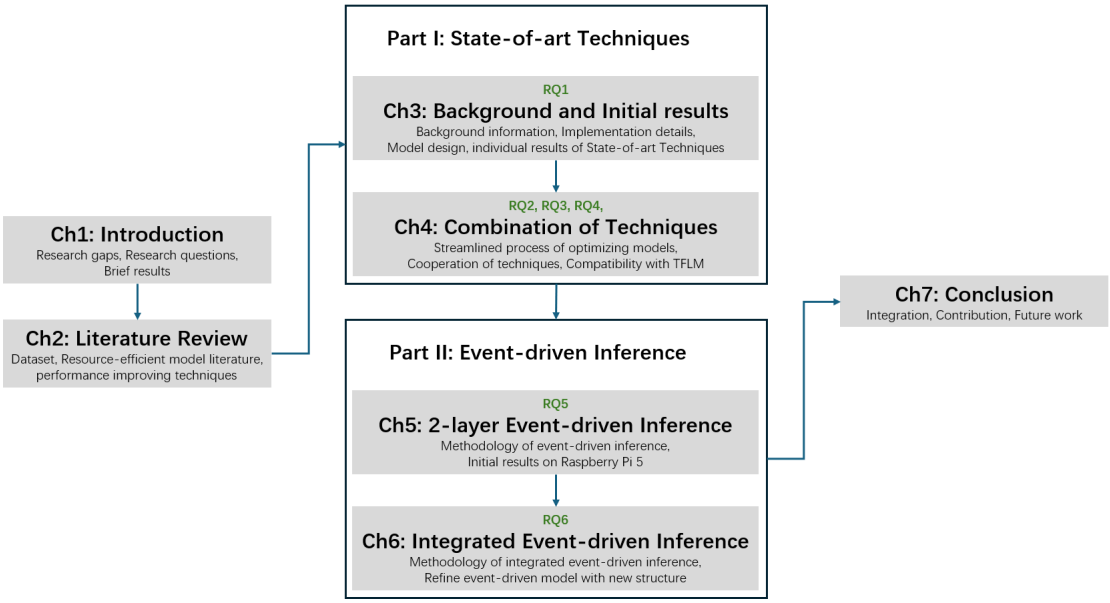

FIGURE 1.1: Thesis outline, including chapters, keywords, and research questions

Chapter 2

Datasets and Existing Techniques

2.1 Datasets

In this thesis, we use four different sensor-based datasets: Opportunity [5], Sensors [20], Wisdm [17], Pamap2 [18]. These datasets differ in several important aspects, including the number of activity classes, the balance of sample distributions, sampling rates, and the types, numbers, and placements of sensors. This variation helps improve the generalizability of our research, enabling us to evaluate model performance across different scenarios and activity distributions.

2.1.1 Opportunity

The Opportunity dataset [5] contains real-world daily activity data collected using various sensors, including accelerometer, gyroscope, and magnetometer. It focuses on body-worn sensor data to classify 4 locomotion activities and 17 micro activities. In our thesis, we excluded the dominant null class, which accounts for over 70% of the data. Data was collected from 4 participants using 5 IMUs placed on the upper body (left/right lower arms, left/right upper arms, and torso), leading to 15 sensors and 45 features. The dataset is very challenging to classify due to highly similar movements, such as opening and closing a door. The sampling rate is 30 Hz.

2.1.2 Sensors

The Sensors dataset [20] includes data for 7 physical activities collected from 10 participants using 5 IMUs placed on various body positions: right and left jeans pockets, right-side belt, right upper arm, and right wrist. Data was collected at a sampling rate of 50 Hz, and this study utilizes all activities with the full set of sensor data at the original sampling rate.

2.1.3 Wisdm

The WISDM dataset [17], a widely used benchmark for sensor-based HAR, was collected using a 3-axis accelerometer in a smartphone placed in the user's pocket to record six daily activities. Data was recorded at a frequency of 20 Hz. Despite its simplicity, the dataset is highly imbalanced, with "jogging" activity comprising around 40% of the data, while "upstairs" accounts for only 1%.

2.1.4 Pamap2

The PAMAP2 dataset [18] contains sensor data from nine participants performing 18 daily activities, ranging from simple tasks like sitting and walking to complex ones like ironing and jumping rope. Data were collected using three IMUs (on the dominant wrist, chest, and ankle) and a heart rate monitor. The IMUs, equipped with accelerometers, gyroscopes, and magnetometers, recorded at 100 Hz, while there is also a heart rate monitor operated at 9 Hz, which is not used in this thesis. This study focuses on 12 common activities across participants, using 27 features from the IMUs to capture detailed movement data.

2.2 Quantization

Quantization is a technique used in deep learning to reduce the size and computational complexity of models by representing weights and activations with lower precision numerical formats, such as integers instead of floating-point numbers, making the model more suitable for deployment on resource-constrained devices like mobile phones or edge hardware. There are different types of quantization, including Float16 quantization, dynamic range quantization, and full integer quantization. Float16 quantization reduces the precision of weights and activations from 32-bit floating-point to 16-bit floating-point. Dynamic range quantization quantizes only the weights to 8-bit integers while keeping activations in their original format, providing faster inference without significant accuracy degradation. Full integer quantization quantizes both weights and activations to 8-bit integers, maximizing computational efficiency and memory savings, making it ideal for deployment on highly resource-constrained hardware.

In this thesis, we primarily focus on full integer quantization, given that many edge devices utilizing TFLM exclusively support 8-bit integer computations. Extensive research has demonstrated that full integer quantization generally boosts the efficiency, meanwhile induces minimal accuracy degradation [3, 7, 10]. While Wu et al. [23] applied full integer quantization to MobileNet and EfficientNet for tasks such as image classification and voice detection, observing accuracy drops ranging from -0.56% to -4.79%. To address these losses, they proposed employing Partial Quantization and Quantization-Aware Training. However, these techniques are insufficient to mitigate the substantial accuracy degradation observed in certain models [12, 19].

Krishnamoorthi [12] reported a drastic decline in top-1 accuracy on the ImageNet validation set, from 70.9% to 0.1%, following full integer quantization. Similarly, Sheng et al. [19] observed a significant drop in MobileNetV1 accuracy on the ImageNet2012 validation dataset, from 70.50% (float pipeline) to 1.80% (8-bit pipeline). They suggested modifying layer designs and enhancing batch normalization to address these issues. However, their experiments were limited to computer vision applications and did not adequately address similar challenges encountered in sensor-based HAR, as observed in our experiments on datasets with relatively higher feature dimensions.

To tackle this challenge, we investigated the application of full integer quantization to simpler models and identified overfitting as a primary cause of accuracy degradation. This thesis explores methods to reduce model complexity, demonstrating their effectiveness in mitigating the accuracy loss associated with overfitting during quantization. These findings offer valuable insights for deploying resource-efficient models on edge devices in sensor-based HAR tasks with multiple sensor inputs. These observations and solutions are thoroughly discussed in Section 3.3.

2.3 Pruning

Pruning helps reduce the size of machine learning models by removing weights that have little impact on performance. Sparsity refers to the percentage of weights in a model that are set to zero. Higher sparsity reduces the number of active parameters, making the model smaller and faster. This process can also make the model simpler and easier to store. Contoli et al. [6] worked on improving Human Activity Recognition (HAR) models by using both quantization and pruning on CNN and LSTM architectures. However, they found that these two methods do not work well together. The main issues were that TensorFlow Lite only supports limited operations, and it cannot handle the sparse matrices created by pruning. Moreover, their experiments only used one dataset, UCI-HAR, which limits how widely their results can apply.

To solve these problems, we explored using quantization with other optimization techniques. This allowed us to create a small yet accurate model that can run on edge devices. We tested this model on the Arduino Nano 33 BLE Sense with all 4 datasets, as explained and demonstrated in Chapter 4.

2.4 Knowledge Distillation & Attention Mechanism

Knowledge distillation is a technique for transferring knowledge from a larger, more complex model (teacher) to a smaller, more efficient model (student). By leveraging the teacher's output, knowledge distillation enables the student to learn not only the correct predictions but also the relative probabilities assigned to all classes, which convey richer information about the underlying data distribution. Two critical hyper-parameters in knowledge distillation are the temperature and alpha. The temperature controls the smoothness of the teacher's output distribution by scaling the logits before applying the softmax function. Higher temperatures produce softer probability distributions, providing the student with finer-grained information about the teacher's confidence across all classes. The parameter alpha determines the trade-off between the distillation loss, derived from the teacher's softened predictions, and the standard loss calculated using the hard labels. This balance ensures the student effectively learns both the nuanced knowledge from the teacher and the direct supervision from ground-truth labels.

Attention mechanisms are used to enhance feature representation by adaptively focusing on the most informative components of the input. The Convolutional Block Attention Module (CBAM) is an efficient and lightweight attention mechanism that combines channel attention and spatial attention to refine feature maps along two complementary dimensions.Channel attention focuses on identifying the importance of each channel in a feature map by using global average pooling and max pooling to capture spatial information. The resulting attention weights adjust the feature map along the channel dimension. Spatial attention highlights important regions within the feature map by pooling across channels and generating an attention map through a convolutional layer, which refines the spatial features.

Knowledge distillation and attention mechanisms have been extensively studied in recent years [8, 13, 25]. Most research has focused on improving model performance, but they often overlook important factors like model size, inference efficiency, as well as the compatibility of TFLM, which are critical for deployment on edge devices. For instance, [9] introduced a dual-attention mechanism to enhance accuracy. However, the increase in model parameters makes it unsuitable for devices with limited resources. Similarly, [2] proposed the Convolutional Block Attention Module to improve accuracy with a relatively low

impact on efficiency. Unfortunately, their testing was conducted only on high-performance CPUs, without considering edge devices. As a result, issues related to compatibility and challenges in deploying these models on resource-constrained platforms were not identified or addressed.

To address these unexplored research gaps, we implemented knowledge distillation and attention mechanisms in the model and evaluated it across four distinct datasets. The model's accuracy, size, and inference time were systematically tested and recorded, as detailed in Section 3.5 and 3.6. Building on these initial results, we applied a combination of the selected techniques, achieving a model that maintained the original accuracy while reducing the size to just 3% of the initial model.

To further examine the compatibility of knowledge distillation and attention mechanisms with TFLM, we successfully deployed the model on an Arduino Nano 33 BLE Sense in Chapter 4. This required specific modifications to the model layers to ensure compatibility. The Feature Selecting technique was also implemented to make the model size fit on Arduino. These experiments not only demonstrated the state-of-the-art advancements in efficiency and performance optimization but also resolved compatibility issues between attention mechanisms and TFLM. Additionally, the work introduced a streamlined approach for developing extremely compact models that achieve relatively high performance, paving the way for practical deployment on resource-constrained edge devices.

2.5 Neural Architecture Search & Dynamic Routing Networks

NAS is a method that automates the design of neural network architectures to find a balance between performance and efficiency [26]. This advanced technique is particularly useful for creating models that can work well on devices with limited computational power. However, NAS requires a lot of computational resources and time during its training phase, as each architecture in the search process needs to be tested and evaluated [24].

DRN is a method used to make neural networks more efficient by simplifying their structure [22]. DRN works by dividing the network into smaller units called cells. Each cell is made up of interconnected points called nodes, and these nodes are linked by different transformation paths. During operation, the network selects only the paths it needs [4]. This approach can help reduce unnecessary computations and improve efficiency. However, the process of selecting paths dynamically adds extra calculations, which can still be challenging for devices with very limited processing power, like microcontrollers [4]. Additionally, having multiple paths in the network increases the amount of storage needed and makes the hardware design more complex [14].

NAS and DRN offer a promising idea for improving inference efficiency on platforms with limited computational capabilities, but both come with significant challenges. NAS requires extensive computational resources during the design phase, while DRN introduces additional overhead during inference due to its dynamic path selection process.

In this thesis, we propose a more practical and feasible technique, introduced in Chapter 5, called event-driven inference. Like DRN, our method dynamically selects inference paths during operation to optimize efficiency, but it eliminates the computational overhead by statically defining the model paths during design time, similar to the approach in NAS.

Specifically, we designed a pre-classifier to categorize input samples into simple or complex classes. Simple samples are processed through a lightweight inference path, consisting of smaller layers, to minimize inference time and energy consumption. Complex samples,

on the other hand, are directed through a more comprehensive path with additional layers and filters to maintain high accuracy.

Additionally, drawing inspiration from [21], which focuses on detecting continuous actions and smoothing output results, we extended our pre-classifier to differentiate between durative and momentary classes. For durative actions, the model avoids reclassifying each sample and only performs pre-classification again when the specific durative action stops, further optimizing efficiency.

In Chapter 6, we discuss integrating this two-layer event-driven structure into a single model. This integration addresses the complexity issues in deploying multiple collaborating models in one inference.

Chapter 3

Background and Initial Results

In this chapter, we present a comprehensive evaluation of the techniques introduced in Chapter 2 through rigorous experimental analysis. These techniques include quantization, pruning, knowledge distillation, and attention mechanisms, each assessed for their effectiveness in optimizing deep learning models for edge deployment.

We first introduce models with a reduced number of filters and LSTM units as benchmarks for comparison in Section 3.2. In Section 3.3, we evaluate the performance of quantization, highlighting its limitations when applied to small models. Additionally, we illustrate potential challenges associated with full integer quantization. By reducing model complexity, we successfully improve the accuracy of the full-integer-quantized model from 33% to 98%. In Section 3.4, we examine the impact of pruning across four datasets, concluding its limitations regarding sparsity and dataset specificity. In Section 3.5, we apply knowledge distillation to the model, observing modest improvements in accuracy, which demonstrates the potential of this approach to enhance the performance of compact models suitable for edge devices.

Building upon recent advancements in sensor-based HAR [2], we incorporate an attention mechanism to further enhance model performance in Section 3.6. While the attention mechanism does not directly contribute to model size reduction, it is well-documented for its capability to significantly improve the accuracy of compact models[2], thereby facilitating efficient deployment in resource-constrained environments.

3.1 Background Information and Implementation Details

The experiments were conducted across three different platforms to evaluate the model's performance and feasibility. Initially, training and preliminary testing were carried out on a laptop equipped with an i9-13900HX processor and an RTX 4080 GPU. This setup provided a robust environment for developing and validating the model. The second phase involved deploying the model on a Raspberry Pi 5 to simulate an edge environment. While the Raspberry Pi 5 is more powerful than typical computationally constrained edge devices, it was selected for its ease of use and to avoid potential compatibility issues with TFLM. This phase allowed for the collection of accuracy and inference time data in a controlled yet edge-like setting. Finally, the model was deployed on the Arduino Nano 33 BLE Sense, a highly resource-constrained device, to assess the challenges of implementing a complex model like DeepConvLSTM on such hardware. This final step highlighted the potential issues developers may encounter, such as memory limitations and processing delays, as well as the compatibility issues, offering valuable insights for real-world deployment on tiny devices.

The experiments were conducted using Python 3.10.0 and TensorFlow 2.10.0. To simplify deployment on the Arduino platform, Edge Impulse [1] was utilized. During performance evaluation, including inference time testing, the batch size was fixed at 128 to fully utilize the testing device's capabilities.

We segment the data into 1-second time windows and use a split ratio of 60% for training, 20% for testing, and 20% for validation, ensuring a balanced approach for model learning, evaluation, and fine-tuning.

3.2 Designing Resource-Efficient Models Using Base Models

We start by introducing the concept of moderate and lite models, which are derived from the original full-size model by reducing the number of filters and LSTM units. Unlike compression techniques like quantization and pruning that make the original model smaller, these lite and moderate models are created simply by decreasing the number of filters, which results in a smaller model size but comes at the cost of some accuracy loss.

As shown in Table 3.1, the models are named LM, MM, MM+, and OM. OM, the original model, has 64 filters in each convolutional layer and 128 units in each LSTM layer. LM, the lite model, has the fewest with 4 filters and 8 LSTM units, while MM and MM+ fall in between these two models.

	Conv Layer	**LSTM Layer**
Lightweight Model (LM)	4 filters each layer	8 units each layer
Moderate Model (MM)	8 filters each layer	16 units each layer
Moderate+ Model (MM+)	16 filters each layer	32 units each layer
Original Model (OM)	64 filters each layer	128 units each layer

TABLE 3.1: Comparison of different model architectures (flipped axis).

The redesigned model structure significantly reduces both model size and inference time by decreasing the number of filters and LSTM units. Although this simplification may lead to a drop in accuracy, it is a necessary trade-off to ensure the DeepConvLSTM model can operate efficiently on edge devices. To address this accuracy loss, additional optimization techniques, such as knowledge distillation (discussed in Section 3.5) and attention mechanisms (discussed in Section 3.6), are applied. The lite and moderate models generated through this redesign are utilized in subsequent experiments for testing, comparison, and meeting specific experimental requirements. For instance, the proposed model in Chapter 4 incorporates a combination of techniques and is based on the MM+ architecture. This choice ensures an optimal balance between accuracy, model size, and inference time, making it the most effective solution for diverse scenarios. In Chapter 5, the LM model is specifically utilized for classifying simple samples due to its ability to achieve precise predictions while maintaining the fastest inference time among all tested models.Although the redesign of the model structure may seem straightforward, it forms the foundation of this thesis. This approach ensures a balanced trade-off between accuracy and efficiency, which is critical for achieving the research objectives

3.3 Quantization

Quantization is a widely adopted technique for deploying machine learning models on edge devices. There are three primary types of quantization: float16 quantization, dynamic

range quantization, and full integer quantization. These methods reduce the precision of model weights and activations to formats such as float16 or int8, thereby significantly lowering memory requirements while maintaining minimal impact on model accuracy.

Most existing research on quantization lacks sufficient emphasis on the distinctions between float16 quantization, dynamic range quantization, and full integer quantization, along with their respective application scenarios. Additionally, the significant accuracy degradation that can occur with full integer quantization when dealing with complex models has not been adequately addressed. This thesis aims to bridge this gap by proposing a viable solution to mitigate the accuracy loss associated with full integer quantization, especially in scenarios involving complex model architectures.

This thesis focuses primarily on full integer quantization, as many edge devices are limited to supporting integer-only inference, particularly with TFLM. The complexities and challenges associated with full integer quantization are discussed in detail, including issues such as reduced accuracy and compatibility constraints. Techniques such as quantization-aware training are proposed as potential solutions to mitigate these challenges [15].

Interestingly, in the experiments conducted, some quantized models demonstrated slightly higher accuracy compared to their original counterparts. This unexpected phenomenon is also observed in some previous literatures [7], with potential explanations such as regularization effects or overfitting in the original models.

The accuracy and model size are depicted in Figure 3.1 and 3.2. With respect to accuracy, both float16 quantization and dynamic range quantization maintain performance levels comparable to the original model. However, in the Opportunity and Sensors datasets, full integer quantization resulted in a significant decline in accuracy. As explained in Chapter 2.1, the Opportunity and Sensors datasets each have 45 feature dimensions, while the Wisdm dataset has 3 and Pamap2 has 27. Higher feature dimensions often make the model's structure and feature distribution more complex. Full integer quantization works by converting floating-point parameters and intermediate results into integer values. However, when the data has many features or uneven distributions, it becomes harder to create a quantization strategy that fits all features and network layers. This difficulty can cause some features to lose critical information during quantization, which introduces errors. These quantization errors are particularly problematic in models with high-dimensional feature spaces because the errors can accumulate and negatively affect classification accuracy. For example, the Sensors dataset may have complex raw feature ranges, distributions, and correlations, which makes it prone to greater errors during full integer quantization. Additionally, certain tasks in Sensors may depend heavily on accurately representing key features. If these important features lose information during quantization, the model's overall performance can suffer significantly. The solution of this accuracy drop will further be explain in this section.

Regarding model size, as shown in Figure 3.2, the results align with theoretical expectations: float16 quantization reduces the model size by approximately half, while dynamic range quantization and full integer quantization achieve reductions to around one-fourth of the original size. Nonetheless, it is important to note that this reduction effect does not always hold consistently for models that are already highly compact. For example, if the number of filters and LSTM layers is reduced to one-sixteenth of the original model, applying quantization may not yield the anticipated reduction of one-half or one-fourth. This phenomenon, as illustrated in Figure 3.3, is likely attributable to fixed overheads, including the network architecture and quantization metadata. Consequently, the benefits of quantization are less pronounced for models that are already highly compact.

Regarding inference time, the results are presented in Figure 3.4. The data is expressed

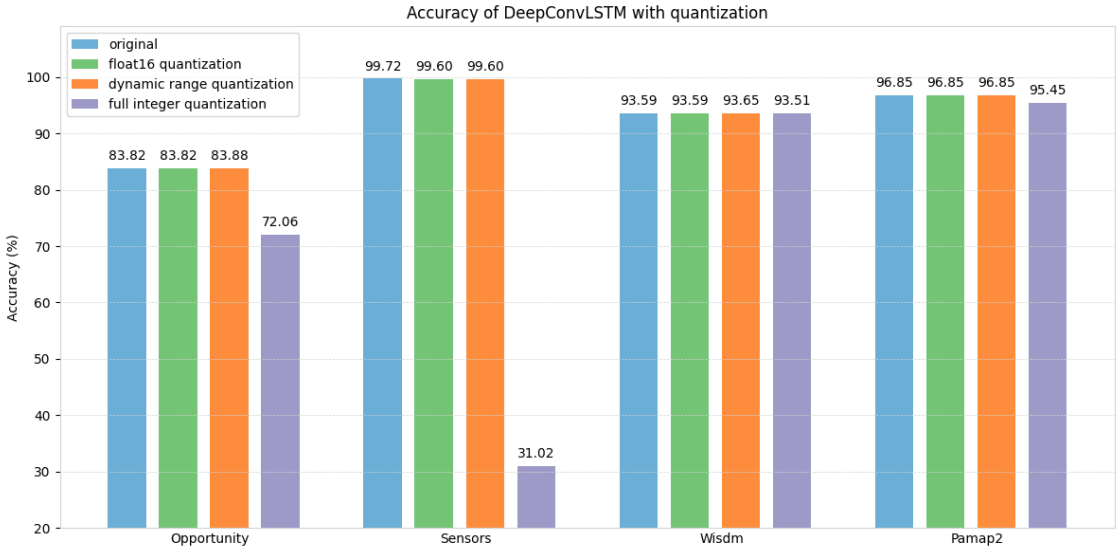

FIGURE 3.1: Accuracy of DeepConvLSTM with quantization

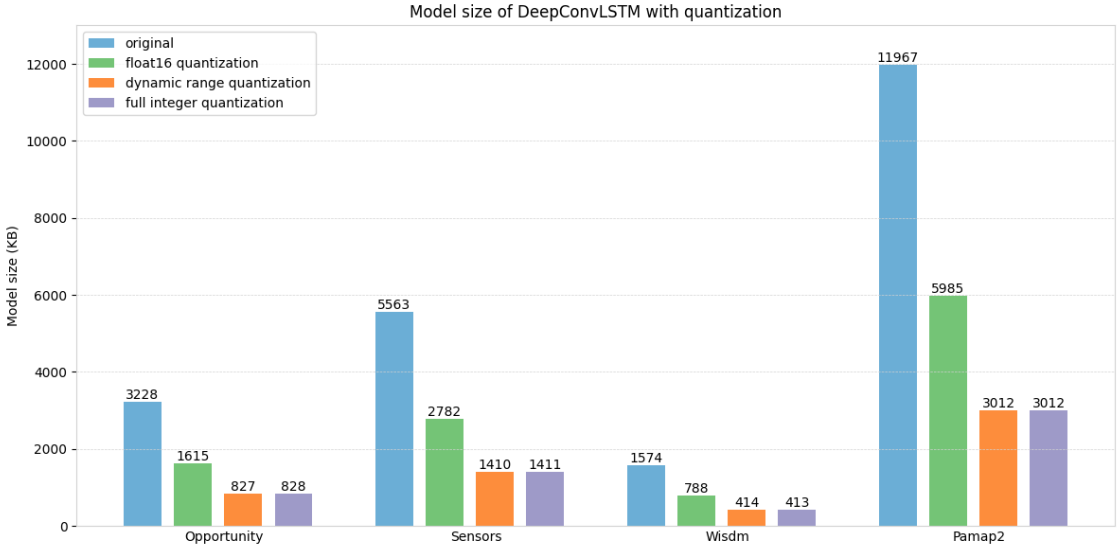

FIGURE 3.2: Size of DeepConvLSTM with quantization

as percentages, with the original model set as the baseline at 100%. This approach was chosen because the inference times vary significantly across different datasets, and using percentages more effectively illustrates the differences of inference time between quantized and unquantized models. For the detailed number of inference time, they are recorded in Appendix 1 and 2.

From the testing, we observed that Float16 quantization has minimal impact on inference time. In contrast, both dynamic range quantization and full integer quantization reduce inference time by approximately 50% across all four datasets. This significant reduction in inference time highlights quantization as an ideal solution for enabling real-time performance of models on devices with limited computational resources.

The DeepConvLSTM model is built with a combination of convolutional and LSTM layers, which makes its structure relatively complex. As discussed before this this sec-

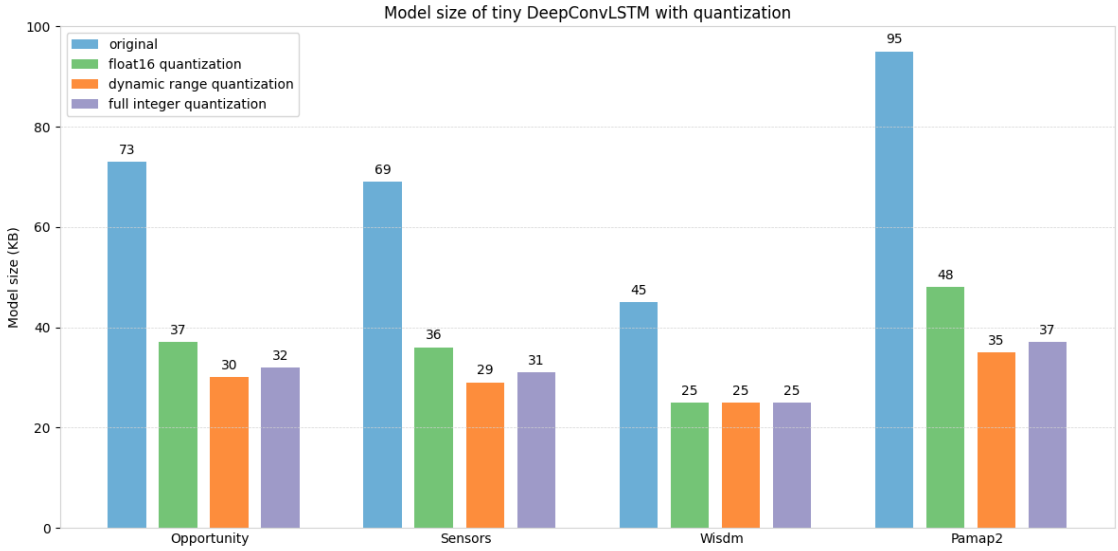

FIGURE 3.3: Size of LM with quantization

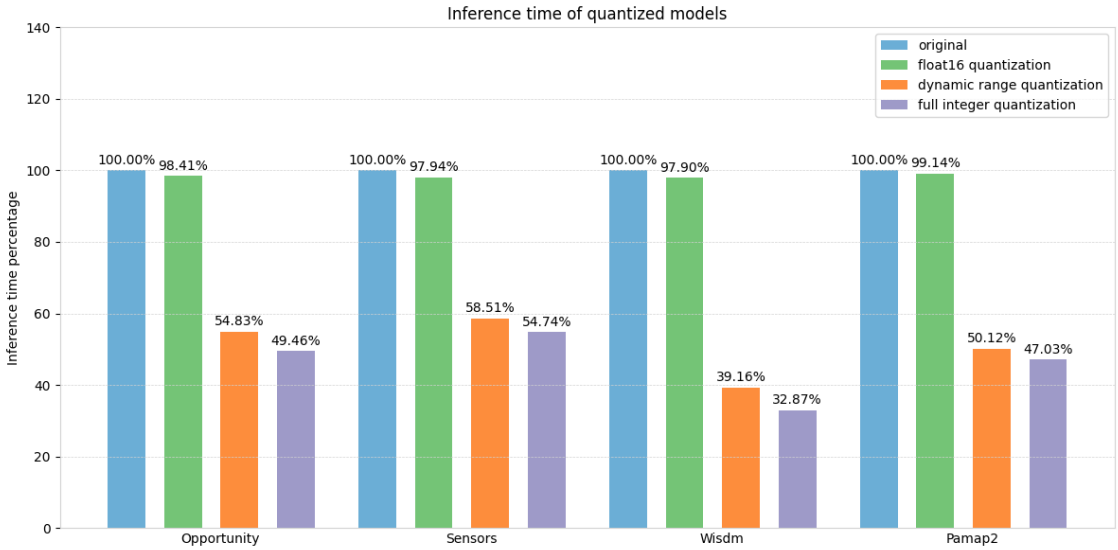

FIGURE 3.4: Time difference of DeepConvLSTM with quantization

tion, when working with datasets that have many features and large parameter scales, like Opportunity and Sensors, quantization errors can accumulate as the data moves through the network. This is because the quantization process, which converts floating-point numbers into 8-bit integers, can lose important details. If the model has redundant features or weights that are unevenly distributed, these errors can grow layer by layer, eventually causing the model's performance to drop noticeably. To address the accuracy degradation associated with full integer quantization caused by overfitting, several approaches were investigated. The most effective methods involved reducing the number of filters, LSTM layers, and the dimensionality of input features, as named as MM and LM model in Section 3.2. As depicted in Figure 3.5, reducing the number of filters and LSTM layers initially led to improved accuracy across both datasets, largely due to the mitigation of overfitting. However, further reductions in model complexity eventually resulted in a

decline in accuracy, indicative of underfitting.

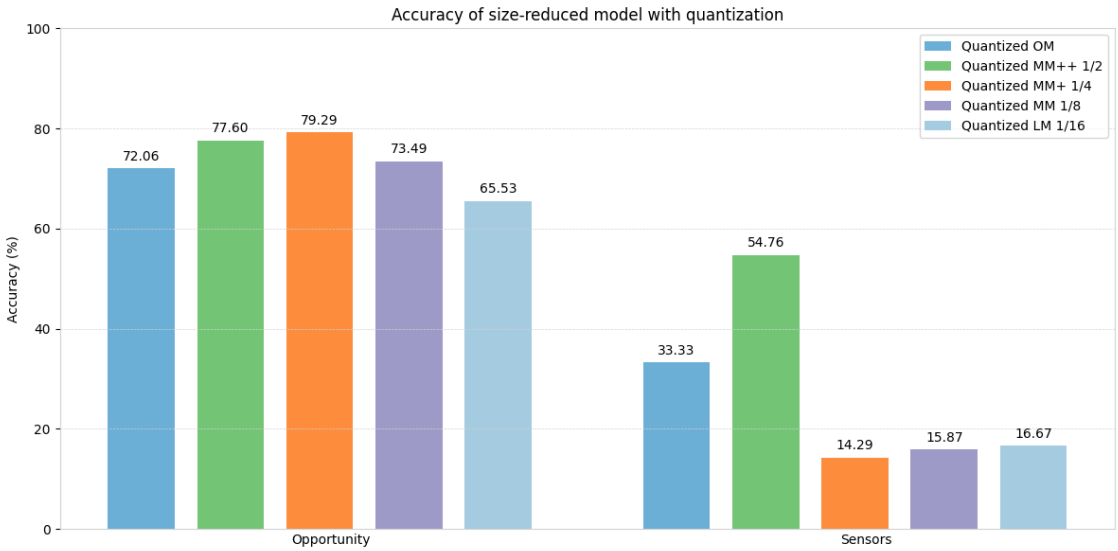

FIGURE 3.5: Accuracy of size-reduced model with quantization

Reducing the number of features can significantly improve the accuracy of full-integer-quantized models to an acceptable level. As discussed in Section 2.1, the Opportunity and Sensors datasets include accelerometer, gyroscope, and magnetometer data collected from five different positions, resulting in a total of 15 sensors and 45 features. In contrast, the Wisdm and Pamap2 datasets have fewer features, with Wisdm containing 3 features and Pamap2 containing 27 features. Given that the Opportunity and Sensors datasets contain more features compared to Wisdm and Pamap2, the models trained on these datasets may suffer from feature redundancy, which can accumulate the accuracy loss resulting from full integer quantization. As illustrated in Figure 3.6, the accuracy of the quantized model actually improves when fewer features are used during training. By evaluating feature importance using SHAP values, the most important half of the features—those contributing to 90% of the total SHAP value—were selected for model training. The results demonstrate that, for both the Opportunity and Sensors datasets, the quantized model trained with fewer features performs significantly better in terms of accuracy, approaching the performance of the original model.

Addressing the accuracy issues associated with full integer quantization is crucial, as it is the only supported quantization format for ultra-constrained edge devices, such as the Arduino Nano, which only supports inference with int8. Optimizing feature selection is therefore key to achieving acceptable performance on these devices.

3.4 Pruning

Figure 3.7 and 3.8 illustrate that pruning has minimal impact on model accuracy when sparsity levels range from 0% to 90%. However, pruning is not always an efficient approach for reducing model size. In our research, pruning significantly decreased model size in the PAMAP2 dataset, while in the Sensors dataset, the model size remained largely unchanged, suggesting that there were few redundant weights in the latter model. It is also important to note that the size comparison was conducted using gzip format, as zero values in h5 and TFLite files continue to occupy memory, resulting in no change in file size before and after

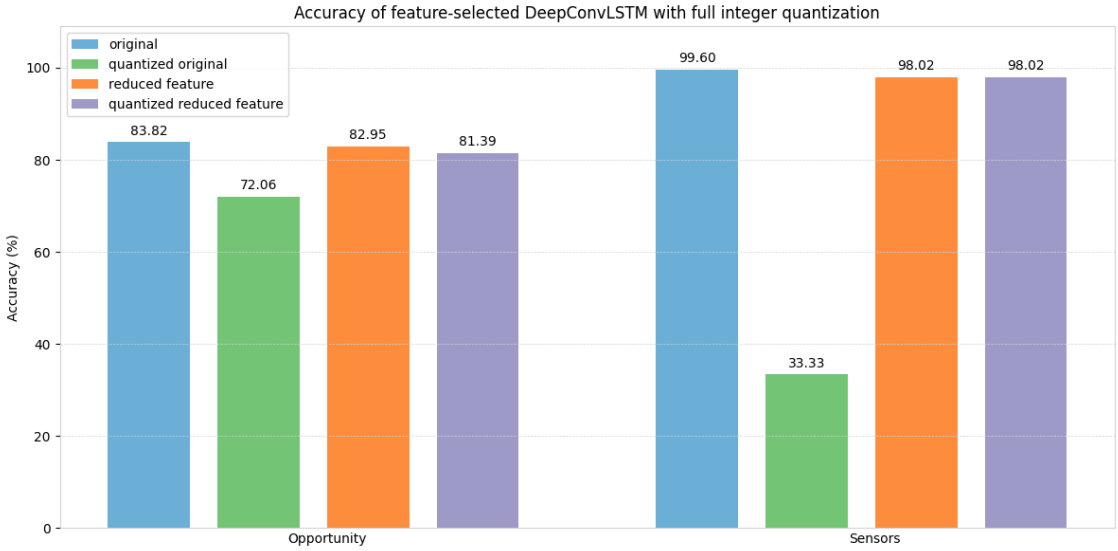

FIGURE 3.6: Accuracy of feature-selected model with quantization

pruning. Because of that limitation, we did not directly compare the size of pruned model and quantized model, while in most of the datasets we tested, quantization performs better than pruning in reducing the model size, except Pamap2. Figure 3.9 shows the inference time differences for the evaluated datasets. Instead of comparing the absolute values, we focus on the time differences to highlight changes across datasets, as the absolute inference times vary significantly. From the results, it is clear that pruning has minimal impact on inference time, with no consistent trend observed. This outcome suggests that pruning does not significantly affect inference time, primarily because hardware without specialized support for sparse matrix operations still processes the zero values introduced by pruning.

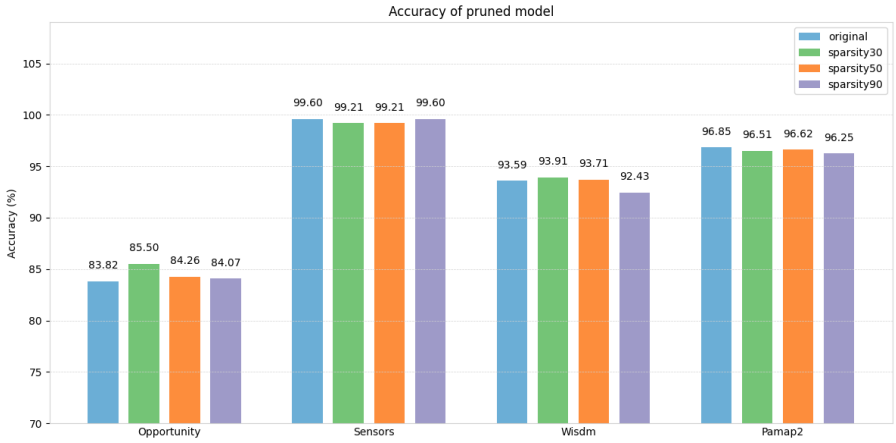

FIGURE 3.7: Accuracy of pruned model

3.5 Knowledge Distillation

Knowledge distillation (KD) is another widely-used technique for training compact models. It is effective in enhancing model accuracy without increasing model size or inference time,

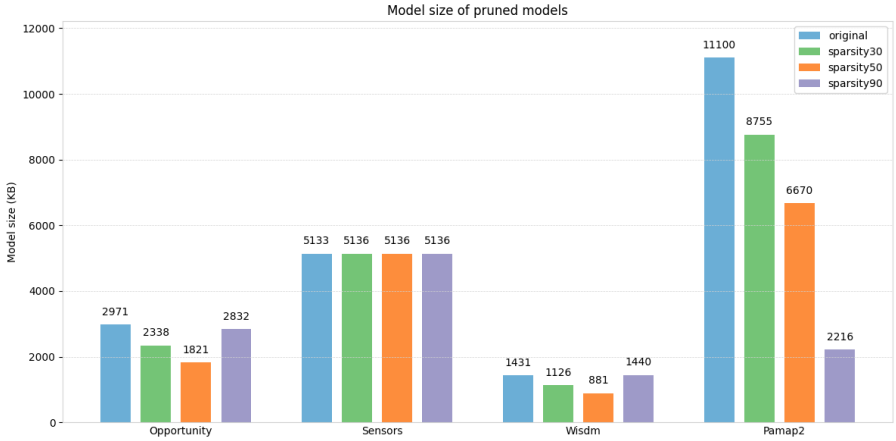

FIGURE 3.8: Size of pruned model

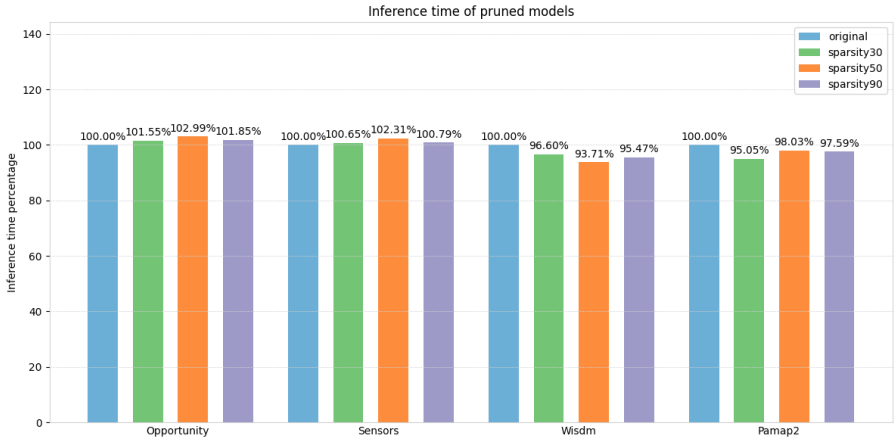

FIGURE 3.9: Inference time of pruned model

making it particularly useful for edge deployment. In our experiments, the optimal student model was identified through a grid search, varying both the temperature and the alpha value to achieve the best performance. In the testing, the original full-size model serves as the teacher, while a MM+ is trained as the student. As discussed in Section 3.3, quantization has limitations when it comes to reducing the size of already small models, such as the LM. Therefore, in this thesis, we chose the MM+ as the student model. The MM+ is large enough that quantization can still significantly reduce its size, making it a better choice for combining with quantization techniques.

Figure 3.10, 3.11, and 3.12 illustrate the accuracy, model size, and inference time of the student model with and without the application of knowledge distillation, respectively. The MM+ KD model achieves similar accuracy to the OM teacher model in the Opportunity, Sensors, and Pamap2 datasets, while also showing a 0.7% improvement in the Wisdm dataset. Regarding model size and inference time, KD does not introduce any changes. Minor variations in inference time were observed, likely due to measurement errors. This demonstrates the effectiveness of knowledge distillation in transferring knowledge from the teacher to the student model. The student model benefits from the higher-level abstractions learned by the teacher without additional computational costs.

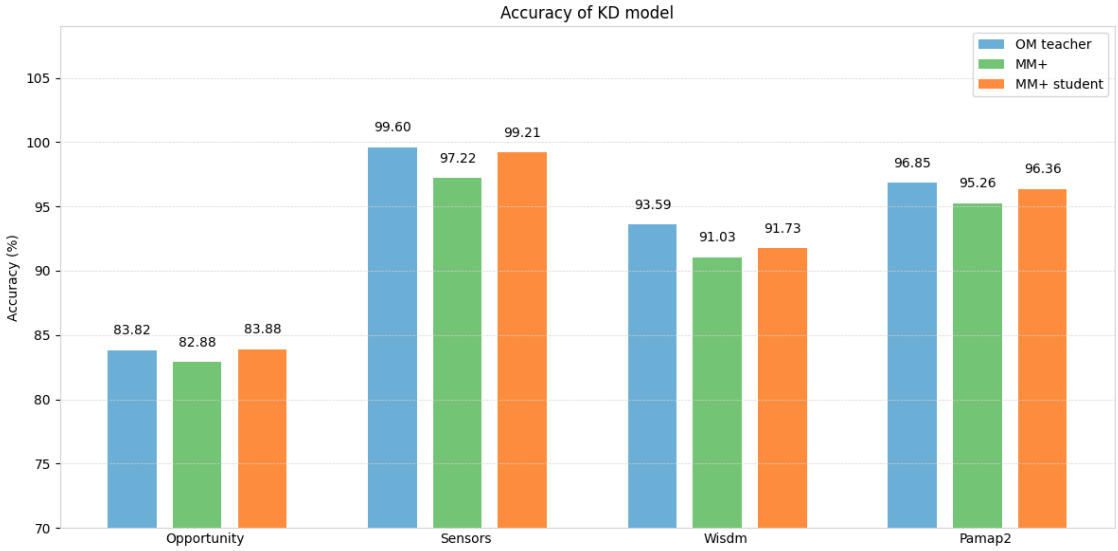

FIGURE 3.10: Accuracy of KD model

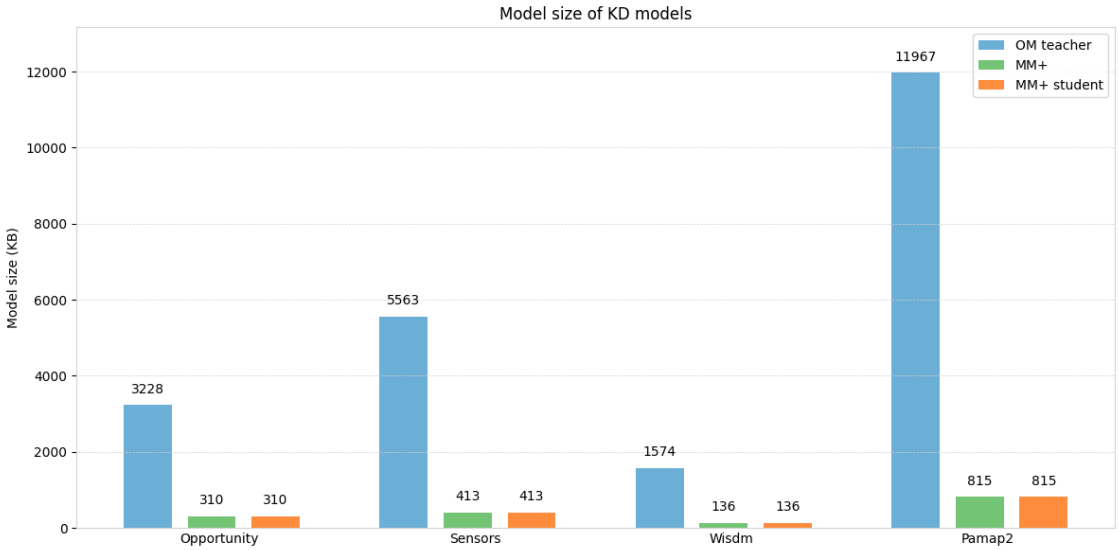

FIGURE 3.11: Size of KD model

3.6 Attention

Attention mechanisms are effective in enhancing model performance by prioritizing the most relevant components of the input. By assigning varying importance to different elements, attention allows models to focus on the most crucial information, thereby improving their capacity to capture complex relationships and effectively manage long-range dependencies. Originally popularized by the Transformer architecture, attention mechanisms have since been widely adopted across various domains, including image recognition and time series analysis. A recent work [2] extended the application of Convolutional Block Attention Module (CBAM) to new fields such as sensor-based HAR, demonstrating its effectiveness in sensor-based machine learning tasks.

In this thesis, attention mechanism is used to recover the accuracy lost due to reducing

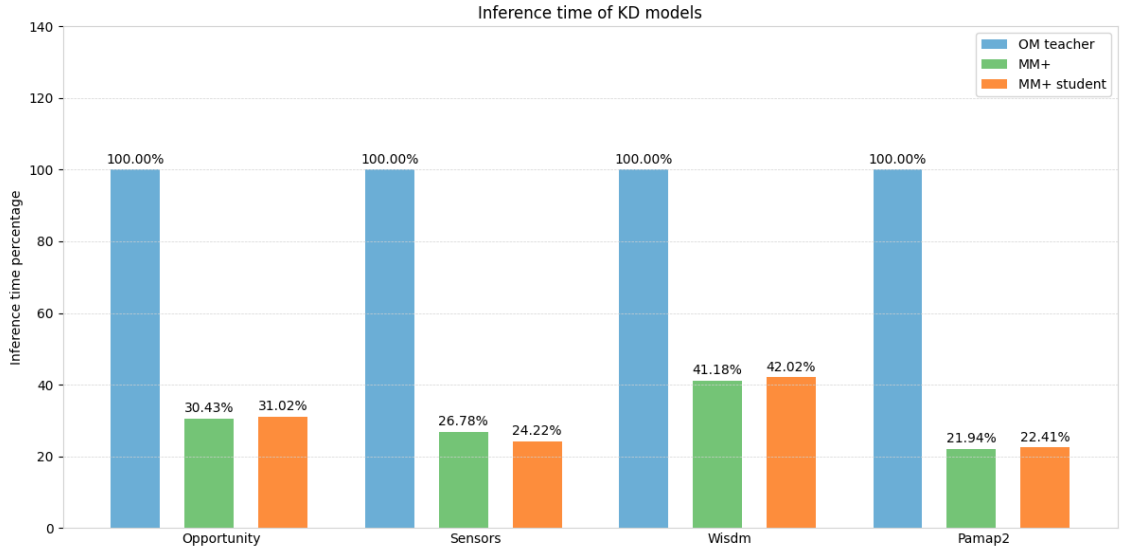

FIGURE 3.12: Inference time of KD model

filters and applying quantization. As shown in Figure 3.13, the attention mechanism improves accuracy across all four datasets. More importantly for edge computing, attention mechanisms have minimal impact on model size and inference time, as demonstrated in Figures 3.14 and 3.15. These results suggest that attention mechanisms can effectively highlight the most important features while reducing the impact of less relevant ones. Although attention mechanisms introduce some computational overhead, the increase is relatively small, making them well-suited for resource-constrained environments. This balance between improving performance and maintaining efficiency makes attention mechanisms an essential component for sensor-based HAR models.Attention mechanisms play a crucial role in this thesis and are applied across multiple models to enhance performance. In Chapter 4, attention is utilized on a quantized MM+ model to mitigate the accuracy loss caused by full integer quantization and structural redesign. Furthermore, in Chapter 6, attention significantly improves the accuracy of the model with feature selection, bringing its performance to levels comparable to or even exceeding that of the OM.

Previous research [2] demonstrated that attention has promising improvements in model accuracy. The interplay between attention mechanisms and other model optimization techniques, such as quantization and knowledge distillation, remains unexplored. This thesis addresses this research gap in Chapter 4 by evaluating the combined effects of attention, quantization, and distillation on lightweight models, thereby assessing their synergistic potential for enhancing performance while maintaining a compact model size.

Besides that, in previous research [2], all experimental evaluations were conducted on an M3 MacBook, which possesses computational capabilities far beyond those typical of edge devices. Consequently, the feasibility of deploying such models on resource-constrained edge hardware was not assessed. This thesis addresses this limitation by successfully deploying the optimized models on the Raspberry Pi 5 and Arduino Nano 33 BLE Sense in Section 4.3. These deployment experiments provide insights into real-world compatibility challenges, and this work presents practical solutions to overcome these issues, ensuring that the models function effectively within the constraints of edge environments.

In conclusion, full integer quantization effectively reduces the model size to one-fourth of its original size, making it suitable for deployment on edge devices running TFLM with

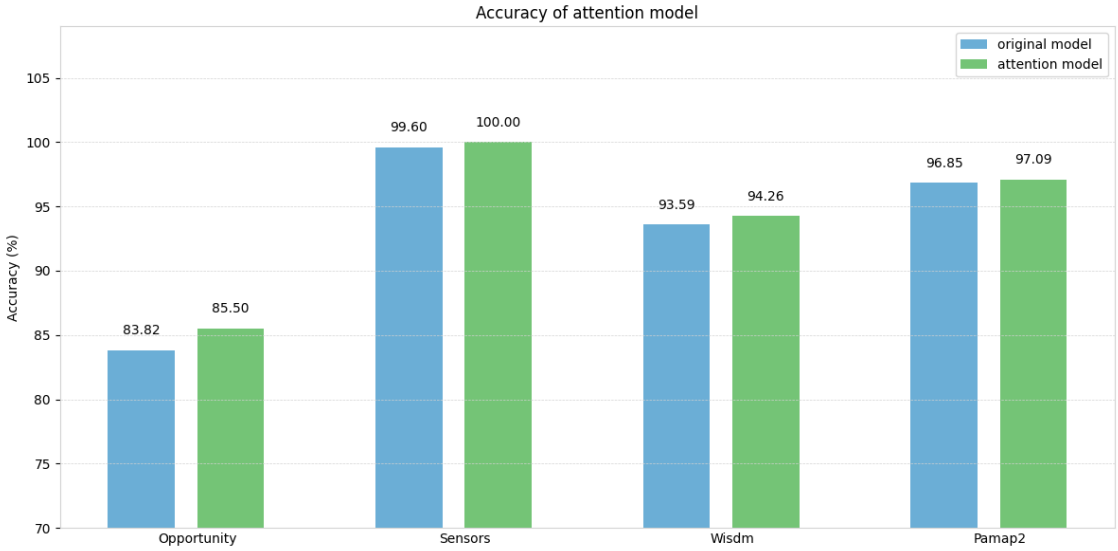

FIGURE 3.13: Accuracy of attention model

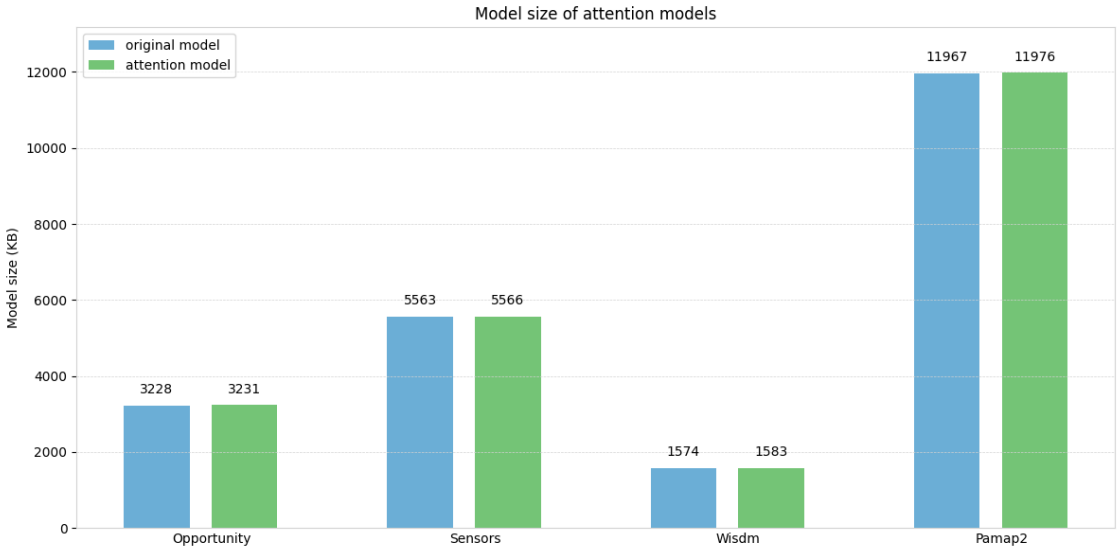

FIGURE 3.14: Size of attention model

int8 inference. However, this process may result in a drop in accuracy. To address this, techniques like knowledge distillation and attention mechanism, specifically the Convolutional Block Attention Module (CBAM) used in this thesis, can be applied to maintain performance. While quantization is a straightforward and efficient method for reducing model size and inference time, it may not always meet the requirements of more demanding applications. Therefore, additional techniques are necessary to further enhance model efficiency and ensure its practicality in resource-constrained environments.

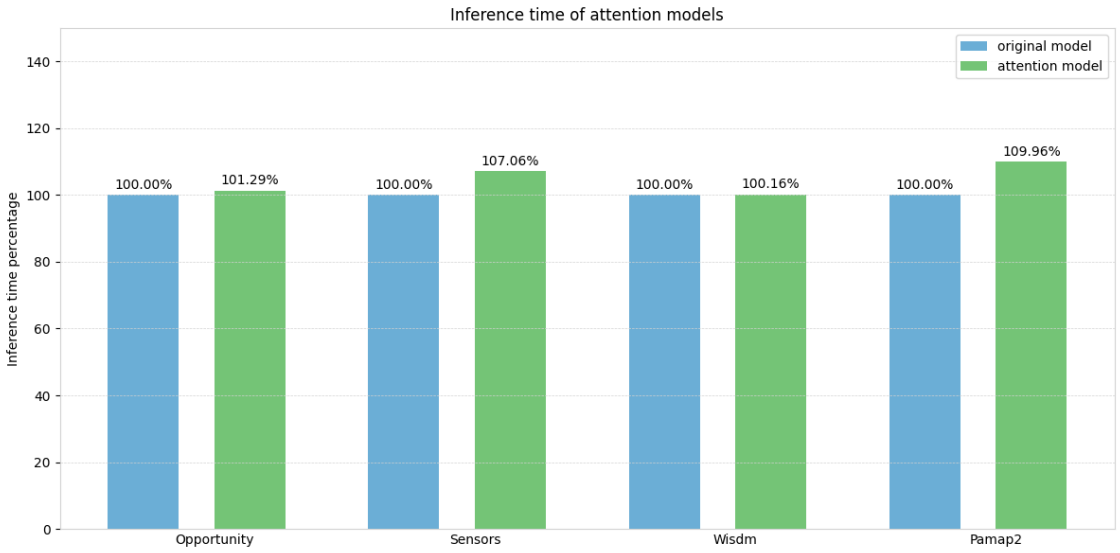

FIGURE 3.15: Size of attention model

Chapter 4

Combination of Quantization, Pruning, Distillation and Attention

In this chapter, we explore the combination of different optimization techniques, including quantization, knowledge distillation, and attention mechanisms, to create an efficient model suitable for deployment on edge devices. First, we validate the model on a PC to ensure compatibility of all these techniques. Then, we deploy the model on an Arduino Nano 33 BLE Sense to demonstrate its feasibility on an edge device and identify potential challenges that may need to be addressed.

By incorporating knowledge distillation, attention mechanisms, and dynamic range quantization, we successfully obtained a compact model that achieves comparable accuracy to the original full-size model while reducing its size to just 3% of the original. In other words, the new model is approximately 35 times smaller. This significant reduction in model size addresses a major challenge in deploying machine learning models on edge devices and demonstrates the feasibility of localized deep learning for sensor-based HAR.

To evaluate the model's performance in a constrained environment, we deployed it on an Arduino Nano 33 BLE Sense, which has only 256KB of RAM. To simplify deployment on the Arduino platform, Edge Impulse was utilized. During deployment, we also resolved compatibility issues with TFLM. Our results demonstrate not only the possibility of localized deep learning with DeepConvLSTM for sensor-based HAR on edge devices, but also the effectiveness of implementing knowledge distillation and attention mechanisms on such constrained hardware. This confirms the practical applicability of these techniques for edge deep learning.

4.1 Methodology

By exploring state-of-the-art technologies in Chapter 3, it has been observed that quantization is one of the most effective techniques for reducing model size while maintaining a high level of accuracy. Although full integer quantization can lead to a reduction in accuracy, this issue can be mitigated by reducing model complexity, making it an essential approach for deployment on integer 8 edge devices. Furthermore, techniques such as knowledge distillation and attention mechanisms are highly effective in compensating for any accuracy loss, particularly when applied to smaller models, with minimal computational overhead.

Given these observations, the approach we proposed for developing a compact and efficient model is illustrated in Figure 4.1. Specifically, the attention mechanism is initially applied to a full-size model, which serves as the teacher model during the knowledge distillation process. After distillation, a smaller student model is produced, and attention

is applied again to enhance its performance. Finally, quantization is applied to the student model to further reduce its size while ensuring compatibility with edge device inference requirements. The performance and results of this model will be discussed in Section 4.2.

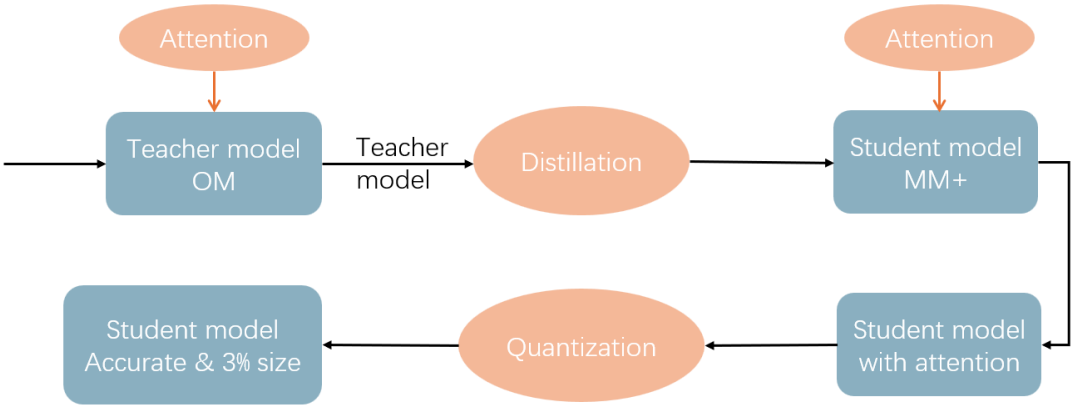

FIGURE 4.1: A solution of combined techniques

4.2 Validation

In this section, state-of-the-art techniques are integrated and evaluated on a Windows laptop to determine potential conflicts or synergies between them. The objective is to assess how these advanced methods interact when applied concurrently, particularly focusing on performance, compatibility, and model size.

Figure 4.2 illustrates that a MM+ model, comprising only 1/4 of the filters and LSTM layers of the original model, can achieve superior accuracy compared to the OM when techniques such as distillation, attention, and quantization are applied. Remarkably, this smaller model is only 3% of the size of the original, demonstrating substantial efficiency gains without compromising performance.

To provide a more detailed analysis, Figure 4.3 offers additional insights. In the lower left quadrant, we observe the performance of LM (a model with 1/16 of the filters and LSTM layers of the original model) and MM (a model with 1/8 of the filters and LSTM layers). While the MM+ model, which incorporates distillation, attention mechanisms, and quantization, achieves a notable improvement in accuracy, ranging from 7% to 10%, compared to LM and MM, but has a similar model size.

Figure 4.2 and 4.3 collectively illustrate that state-of-the-art techniques such as distillation, attention, and quantization can be effectively integrated, resulting in significant performance enhancements even for models with significantly reduced complexity.

4.3 Application

Deploying a DeepConvLSTM model on compact devices, such as the Arduino Nano, presents several compatibility challenges. These challenges primarily arise from the limitations of the hardware, which only supports int8 inference and TFLM, a subset of operations from TensorFlow Lite. One major issue is that LSTM layers introduce operations involving loops (e.g., "while" operations), which are not supported by TFLM. To address this

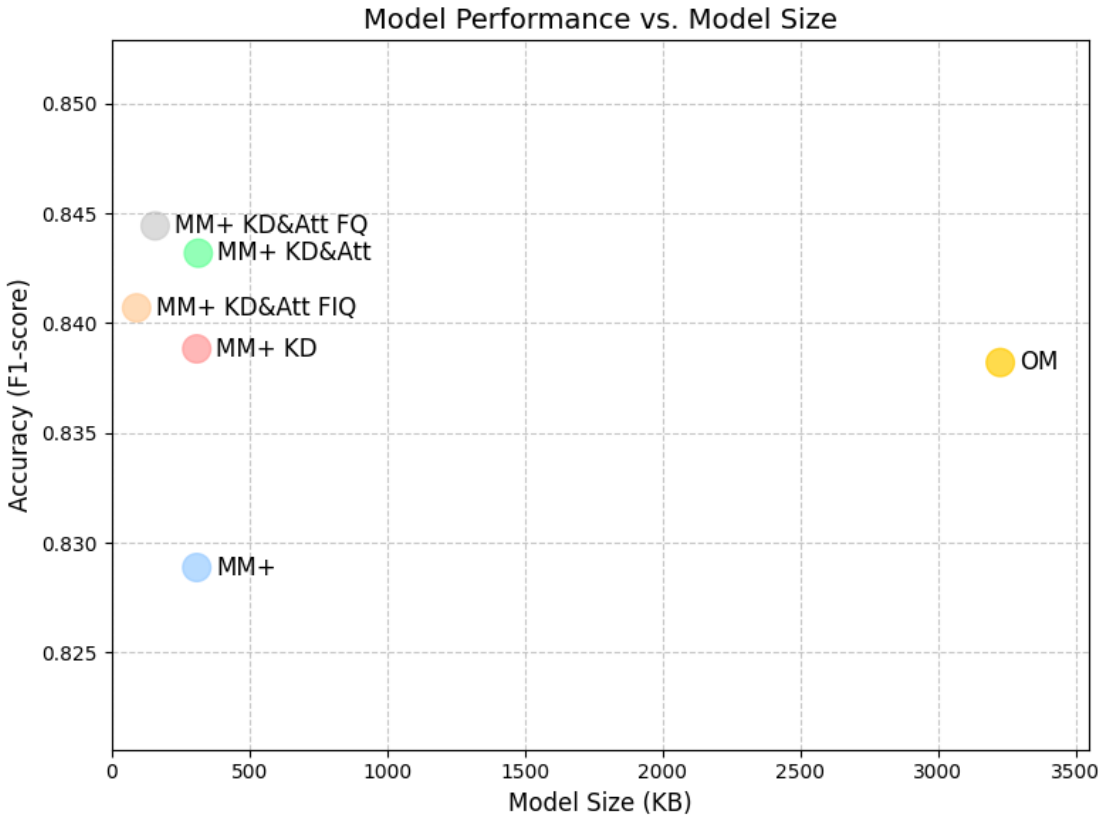

FIGURE 4.2: Performance vs. size

limitation, LSTM layers are flattened, effectively transforming the sequential processing into a form that TFLM can handle. However, this flattening significantly increases the model size. For instance, in the case of the Opportunity dataset, the model size expanded from 45 KB to 431 KB, making it unsuitable for deployment on the limited RAM of the Arduino Nano.

Moreover, mentioned in Section 3.3 that full integer quantization, which is necessary to convert the model for int8 inference, often results in substantial accuracy degradation. Given these two challenges of increased model size and potential accuracy loss, feature selection was applied as an effective solution. By selectively retaining the most relevant features, the model size was reduced to 218 KB, making it feasible for deployment on the Arduino Nano.

The goal of this thesis to deploy the model enhanced with knowledge distillation and attention mechanisms on the Arduino Nano 33 BLE Sense. The challenge we encountered involved the compatibility of TFLM. Specifically, the channel attention and spatial attention mechanisms introduced operations that were unsupported by TFLM, such as the use of the Reshape function within average pooling and max pooling, which led to dynamic output tensor shapes. To address this, we manually implemented the global average pooling and max pooling operations, thereby resolving the compatibility issues.

To deploy the model on the Arduino Nano, we utilized Edge Impulse to simplify the process significantly. By uploading the desired model, the platform automatically checks for compatibility, ensuring the model is fully integer-quantized and free of unsupported operations. Once validated, Edge Impulse generates a deployment package tailored to the target device. We can then upload this package to the Arduino IDE for deployment onto

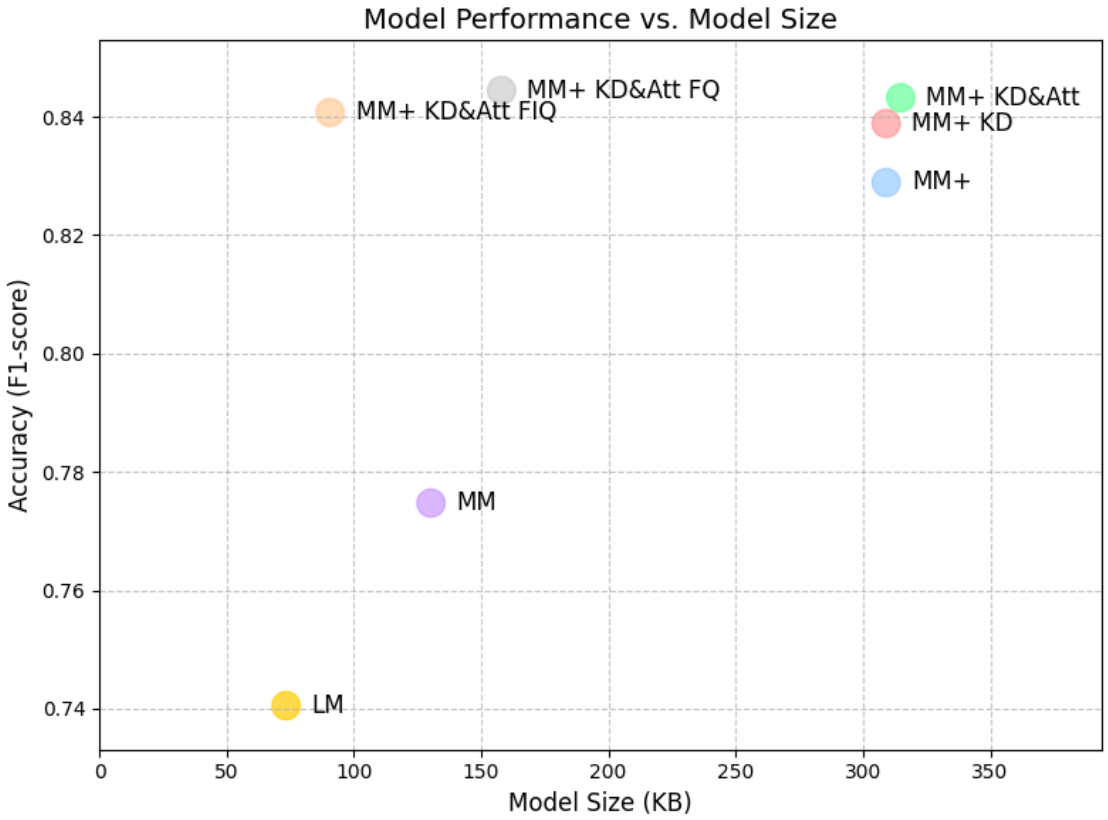

FIGURE 4.3: Performance vs. size

the Arduino Nano. After deployment, the model runs on the device and outputs predictions through the terminal. Then input is sent manually by us as datasets are used instead of real-time inference.The entire process of deployment with Edge Impulse is seamless and straightforward, making it an efficient solution for edge device deployment.

In our thesis, we successfully deployed our model with knowledge distillation and attention mechanisms on the Arduino Nano 33 BLE Sense. The model operated as intended, providing identical outputs to those observed during validation on the initial platform. This accomplishment demonstrates the feasibility of running a DeepConvLSTM model enhanced with knowledge distillation and attention mechanisms on a highly resource-constrained edge device.

Dataset	**Inference Time per Sample (ms)**	**Model Size (KB)**	**used RAM (KB)**	**used Flash (KB)**	**Accuracy (%)**	**Accuracy of OM (%)**
Opportunity (selected feature)	813	264	102.8	182	79.34	82.76
Sensors (selected feature)	979	332	140.9	221.2	98.02	99.21
Wisdm	83	58	17	98.8	91.82	92.17
Pamap2	2531	428	182.6	326.4	94.41	96.67

TABLE 4.1: Deployed model with attention and KD on Arduino

As shown in Table 4.1, the optimized models operate successfully on the Arduino Nano 33 BLE Sense, which has only 256 KB of RAM.To measure how long the model takes to make a prediction, the script records the time it starts and finishes. The time difference is shown directly in the terminal. For datasets such as Opportunity, Sensors, and Wisdm, real-time processing is achieved. However, for Pamap2, processing a 1-second sample

requires 2.5 seconds due to the dataset's high sampling rate and large number of features, highlighting the impact of dataset complexity on inference performance.

In terms of resource usage, the numbers of used RAM and flash are the predicted value provided by Edge Impulse. As we can see in Table 4.1, all models fit within the Arduino Nano's constraints, with RAM usage ranging from 17 KB (Wisdm) to 182.6 KB (Pamap2), and Flash usage remaining well below the device's 1 MB limit. This demonstrates the models' suitability for deployment on highly resource-constrained hardware while leaving room for potential extensions.

Regarding accuracy, the models exhibit minimal degradation compared to the OM. The largest accuracy drop, 3.42%, occurs in the Opportunity dataset, primarily due to the reduction in features and the dataset's inherent complexity. Meanwhile, the Wisdm dataset achieves an impressive inference time of just 83 ms, showcasing the efficiency of the optimized models on simpler datasets. These results illustrate a balance between resource efficiency and performance across various datasets.

Overall, this process can be considered a streamlined method for deploying complex models on highly resource-constrained edge devices. The experiments further validate the feasibility of using a DeepConvLSTM model on the Arduino Nano 33 BLE Sense with minimal performance trade-offs, emphasizing its practicality for real-world applications.In future research, we plan to deploy this model on the OpenEarable platform to enable real-time HAR. OpenEarable is a wearable device designed to be worn on the ear and is also powered by the Arduino Nano 33 BLE Sense. The integration of DeepConvLSTM in this field significantly enhances the model's capability to classify complex activities with greater accuracy and efficiency.

Chapter 5

Event-driven Inference

While the efficiency improvements achieved in Chapter 4 using state-of-the-art techniques are significant, these methods are general solutions applicable to a wide range of tasks. However, in the field of HAR, there are additional opportunities to enhance efficiency by analyzing the specific characteristics of HAR tasks. For instance, not all sensors or input data are necessary at all times, allowing for selective utilization to optimize resource use and performance.

In this chapter, we introduce the concept of event-driven inference, a method designed to improve computational efficiency by adapting the model's complexity based on the input data. This approach aims to reduce inference time significantly while maintaining sufficient accuracy. The proposed method was tested and evaluated on a Raspberry Pi 5 to simulate an edge computing environment. This device was chosen because deploying multiple models in a single project on an Arduino involves considerable challenges because of the limited RAM and lack of support of dynamic execution.

Experimental results indicate an approximate 50% reduction in inference time compared to the original model, without any significant loss in accuracy. Moreover, after applying quantization, the event-driven model further reduces inference time to approximately 35% of the original quantized model. Then, the optimal MM+ generated in Chapter 4 is used to replace the OM to further reduce the model size and inference time.

To enhance the model's performance, we applied a combination of advanced techniques from Chapter 3 and 4, including quantization, knowledge distillation, and attention mechanisms. These methods allowed us to achieve a highly efficient model while maintaining relatively high accuracy, making it suitable for deployment in resource-constrained environments.

5.1 Methodology

To further reduce inference time, the concept of event-driven inference is introduced. In the context of HAR applications, the goal is to accurately detect activities such as going upstairs, going downstairs, and jumping. In these scenarios, sensors such as accelerometers, gyroscopes, and magnetometers are all crucial for classifying these complex actions. However, users may spend a large proportion of time engaged in simpler activities like sitting, standing, or walking. During these simpler actions, not all sensors or features are necessary for accurate classification, and reducing the number of active sensors can save both sensor energy and computational power on the MCU, as well as reduce overall inference time.

Based on these insights, event-driven inference is proposed, as illustrated in Figure 5.1

and 5.2. In Figure 5.1, the input data is initially pre-classified into simple and complex categories. If the sample is classified as a simple activity, a smaller model with a limited set of features is employed for inference, enabling faster computation with reduced resource consumption. Conversely, if the sample is identified as a complex activity, the OM or the optimal MM+ is used to ensure accurate classification of these more challenging tasks. This approach aims to achieve an optimal balance between efficiency and accuracy, tailoring the computational load based on the complexity of the activity being performed.

Based on the analysis of the confusion matrix, we categorize the classes into simple and complex groups. Simple classes are those that the LM can predict with high accuracy. For these classes, the LM is employed to ensure faster inference. On the other hand, complex classes are those where the LM achieves less than 80% accuracy. To maintain high accuracy for these predictions, an OM or optimal MM+ is used instead.

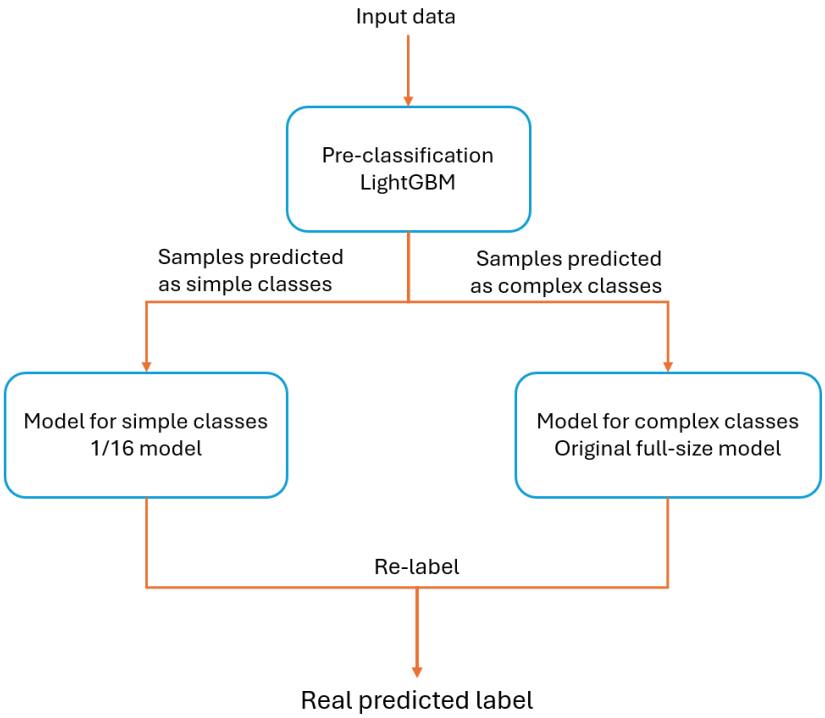

FIGURE 5.1: Event-driven inference with simple/complex

An alternative classification approach involves categorizing activities into momentary and durative classes, as illustrated in Figure 5.2. For example, detecting a user's jogging, jumping, and squatting actions can benefit from this categorization. Jumping and squatting are momentary actions and might be difficult for a model to distinguish due to their short duration and similar features. Jogging, on the other hand, typically lasts longer and is classified as a durative action. During jogging, the model does not need to have the ability of differentiating between jumping and squatting, allowing the use of a simpler model to enhance inference efficiency.

In this thesis, we implement this idea by introducing a pre-classification model to determine whether an action is durative. If a durative action is detected, a lightweight LSTM model is deployed to monitor the activity. Once the LSTM model predicts the end of the durative activity, the system reactivates the pre-classification model to check if

a momentary action occurs. If a momentary action is detected, a more computationally intensive model is employed to ensure high accuracy.

By using this strategy, the more computationally demanding model remains inactive until a momentary action is identified. This approach conserves energy and processing time, making it especially effective in edge computing environments where resources like energy and computational power are limited. The proposed system balances efficiency and accuracy, ensuring optimal performance for real-world applications.

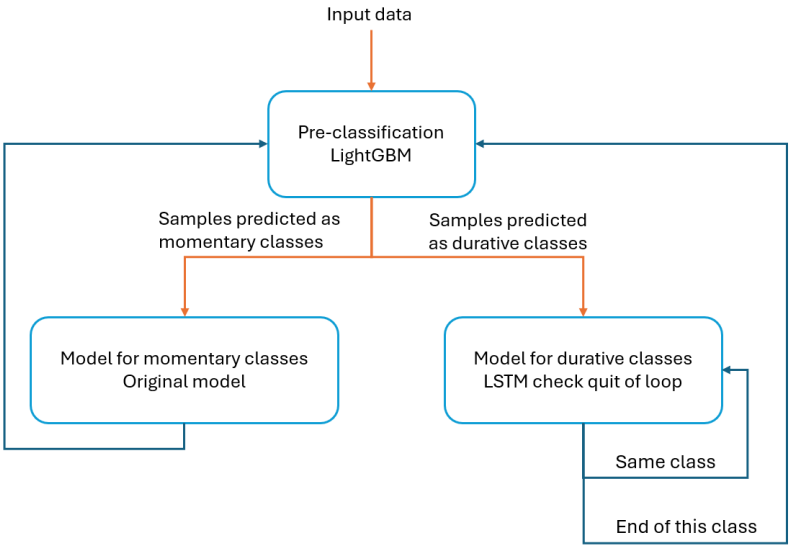

FIGURE 5.2: Event-driven inference with momentary/durative

This alternative classification approach is inspired by [21]. Shoaib et al. [21] proposed a hierarchical lazy smoking detection algorithm that smooths output results by examining both previous and subsequent samples to correct undetected smoking events. Building on this concept, we applied a similar idea to identify the end of durative actions. This thesis extends prior work by investigating two distinct event-based inference strategies: (1) categorizing samples as simple or complex activities, and (2) distinguishing between momentary and durative actions. Both methods have been implemented and thoroughly tested on edge devices. Additionally, in Chapter 6, this research introduces a novel application of event-based inference within the DeepConvLSTM architecture by selectively activating specific layers based on the type of activity being inferred. The study provides a detailed evaluation of the accuracy and inference time associated with these methods, offering key insights into their feasibility and effectiveness for deployment on resource-constrained devices.

5.2 Validation and Application

In this section, we apply and test the event-driven inference method on the Raspberry Pi 5. We primarily compare the accuracy and inference time of this method with those of the original model to evaluate its effectiveness.

The event-driven inference approach was initially implemented on the WISDM dataset. As simple samples account for a significant portion of the overall dataset in Wisdm, specifically 79%, it is considered as an ideal starting point for testing the event-driven inference

framework. Among all the classes, jogging and walking are defined as simple class as LM can precisely predict them with more than 95% of accuracy. The rest of the classes including upstairs, downstairs, sitting, and standing are classified as complex type.

For the pre-classification stage, LightGBM was selected due to its lightweight architecture and high accuracy in binary classification tasks. The accuracy of the pre-classification is 95.08%. We set the minimum baseline accuracy for pre-classification at 95% because the reliability of event-driven inference heavily depends on the accuracy of pre-classification. While a higher accuracy is preferable, it is not essential, as 95% accuracy is sufficient to ensure the feasibility of event-driven inference. Additionally, we considered the trade-offs between model size and inference time. In the subsequent tests in Chapter 5 and Chapter 6, we consistently maintain this baseline and often achieve way higher accuracy during pre-classification. As illustrated in Figure 5.3, event-driven inference results in a slight decrease in accuracy compared to the original model. However, the difference remains minimal, irrespective of whether quantization is applied. When using optimal MM+ obtained from Chapter 4 with knowledge distillation and attention, for the complex samples, a drop of 3.73% is observed.

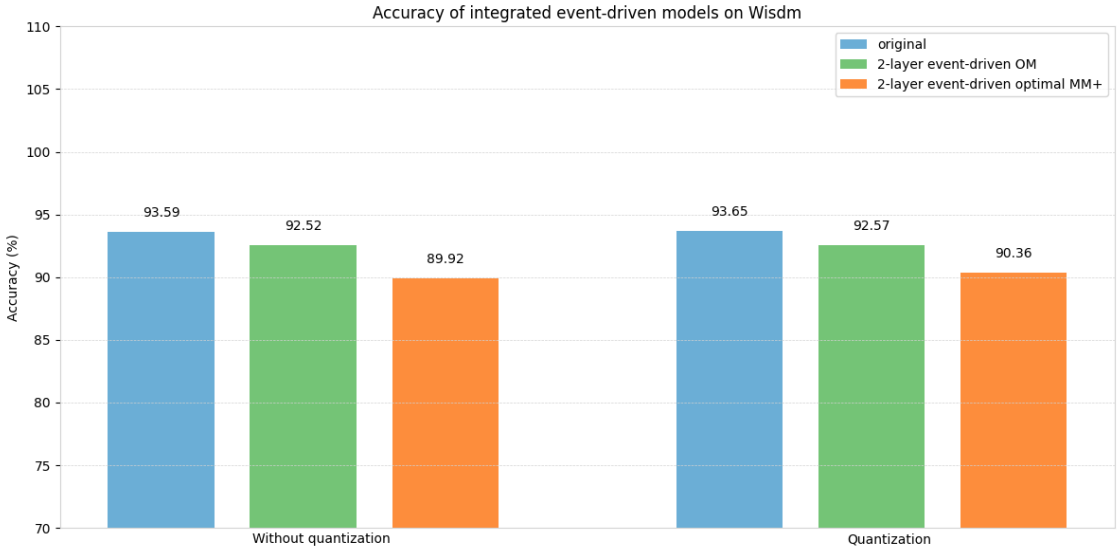

FIGURE 5.3: Accuracy of event-driven model on Wisdm

Importantly, event-driven inference significantly reduces inference time, as evidenced by Figure 5.4. Before quantization, the event-driven approach reduced inference time by approximately 50%. In the quantized model, the event-driven inference method further reduced the inference time to 30% of that of the quantized OM. In the case of event-driven model with optimal MM+, the inference time is even reduced to 13.7%.

However, Event-driven Inference introduces additional complexity to the model size, as it includes two extra components: the LM and a LightGBM pre-classification model. This addition increases the overall size of the system as shown in Figure 5.5. When quantization is applied, the issue becomes even more pronounced. Specifically, the event-driven model is 50.6% larger than the OM. While in the case of using optimal MM+ instead, the overall size is smaller than the OM, but this is mainly because MM+ is significantly smaller than OM. To address this limitation, an integrated event-driven model was introduced, as detailed in Chapter 6. This new design combines the components into a unified structure, reducing redundancy and improving efficiency without compromising performance.

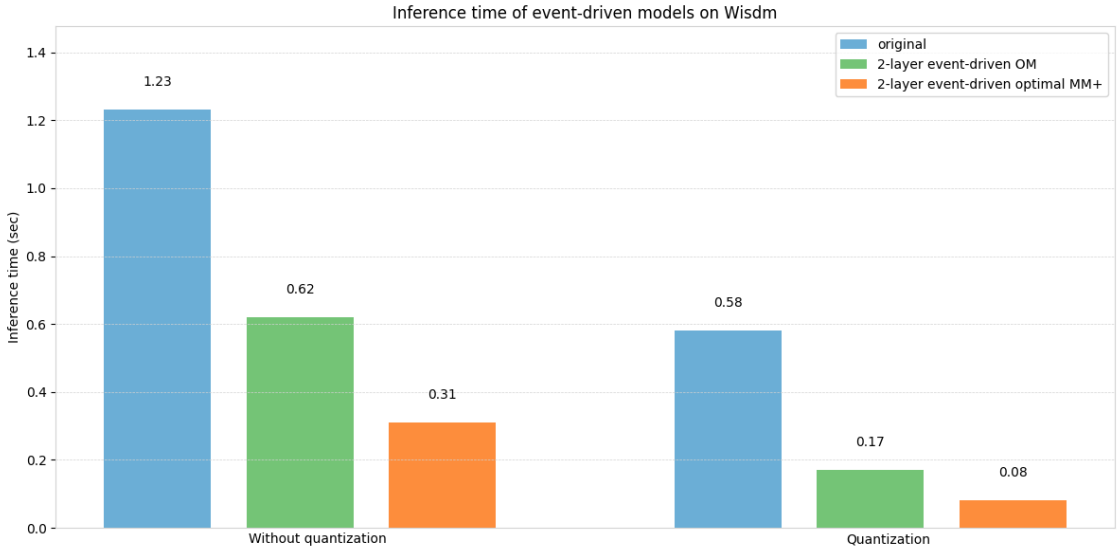

FIGURE 5.4: Inference time of event-driven model on Wisdm

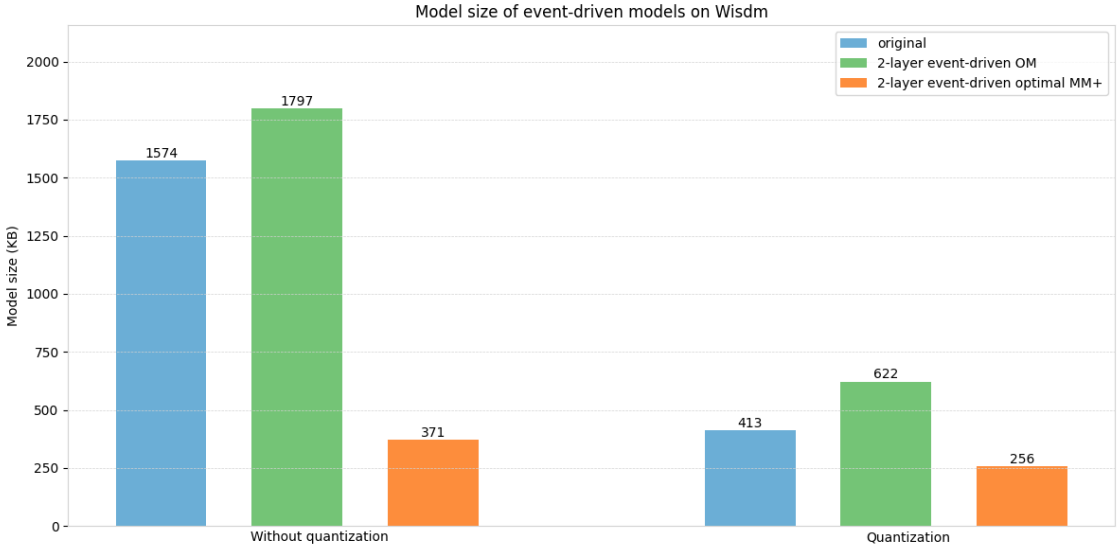

FIGURE 5.5: Model size of event-driven model on Wisdm

To further investigate the effectiveness of event-driven inference, it was combined with feature selection methods to optimize performance. During the pre-classification and classification of simpler samples, not all input features are necessary. By reducing the number of features used during these simpler inferences, additional reductions in inference time can be achieved. To evaluate this approach, the Opportunity dataset was selected, as it contains 45 features derived from 15 different sensors on 5 different positions. Feature importance was first assessed using SHAP values, then 22 out of the 45 features were retained as they contributed to 90% of the total SHAP value. This feature-selecting mechanism is not applied before on Wisdm because it only contains 1 sensor/3 features in the whole dataset, making selecting feature not a feasible method. In the Opportunity dataset, clean table and drink from cup are marked as simple classes. And the remaining classes including 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, toggle switch are complex classes.

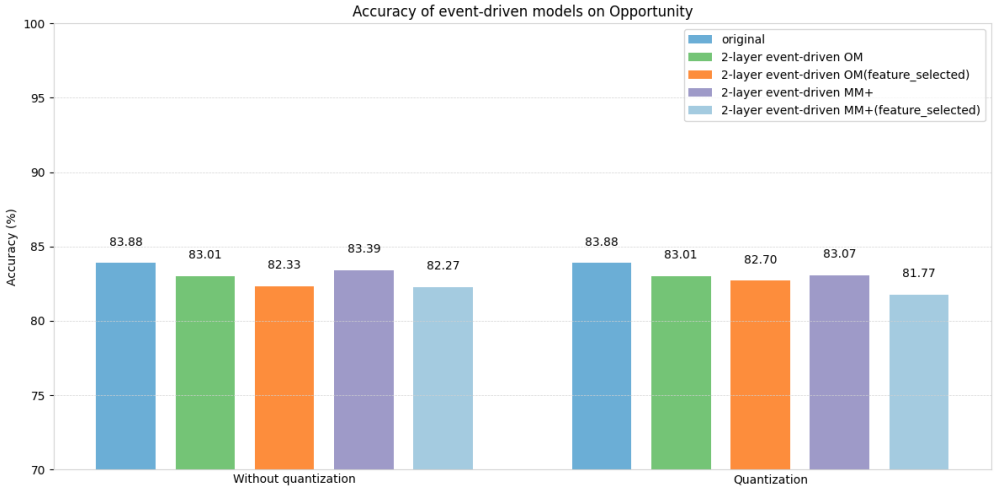

FIGURE 5.6: Accuracy of event-driven model on Opportunity

The results, depicted in Figure 5.6, show that the integration of the event-driven inference framework with feature selection has a negligible negative impact on model accuracy. Both quantized and unquantized models maintained similar levels of performance in the case of using OM or optimal MM+, indicating that the accuracy trade-off is minimal. In the case of Opportunity dataset, the accuracy of pre-classification is 98.63%.

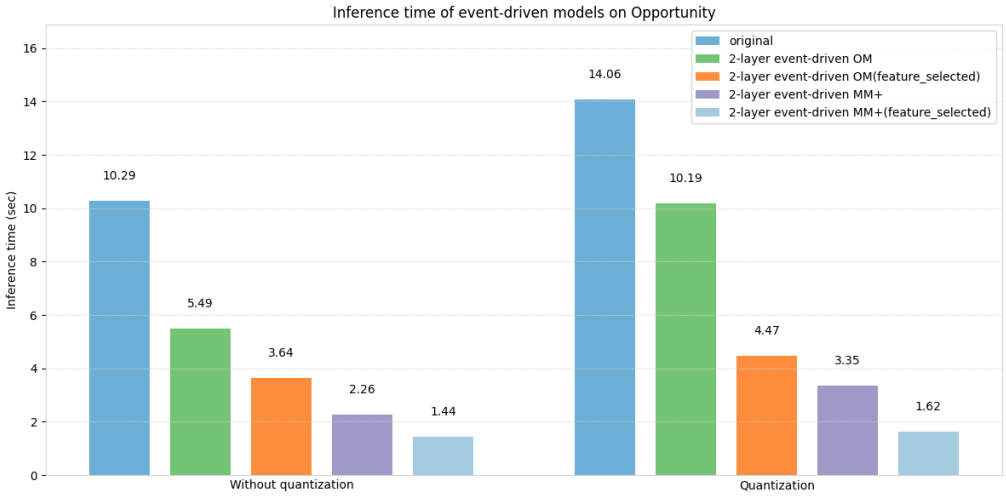

FIGURE 5.7: Inference time of event-driven model on Opportunity

In terms of inference time, the event-driven inference strategy significantly reduced computation time, which was further improved when combined with feature selection as shown in Figure 5.7. Specifically, prior to quantization, the event-driven model alone reduced inference time by approximately 50%. When integrated with feature selection, inference time was further reduced to 35% of the original. After replacing the OM with optimal MM+, the inference time is even reduce to 22% and 14%.

Interestingly, for the Opportunity dataset, the graph shows that the model increased inference time after quantization. This is primarily due to TFLite limited optimization for

LSTM layers. In our testing, TFLite models generally required more time for inference compared to Keras models. This difference arises because certain operations within the LSTM layer are not as fully optimized in TFLite as they are in Keras's native CPU kernels. Although quantization successfully reduced inference time by approximately half, as shown in Figure 3.4, the structural inefficiencies of the TFLite model on DeepConvLSTM caused it to take more time than the original Keras model. Interestingly, this issue was not observed in the Wisdm dataset. Due to its lower sampling rate and fewer features, the computational demands on the LSTM layers were significantly reduced, resulting in better performance for TFLite models. While DeepConvLSTM does not perform optimally in TFLite format, the quantized event-driven model, combined with feature selection, effectively addressed this limitation. This approach reduced inference time to just 32% of the original duration, demonstrating its potential to mitigate the inefficiencies of TFLite in handling LSTM operations. After replacing the OM with optimal MM+ for complex samples, the inference time is further reduced to 11.5% of the quantized OM.

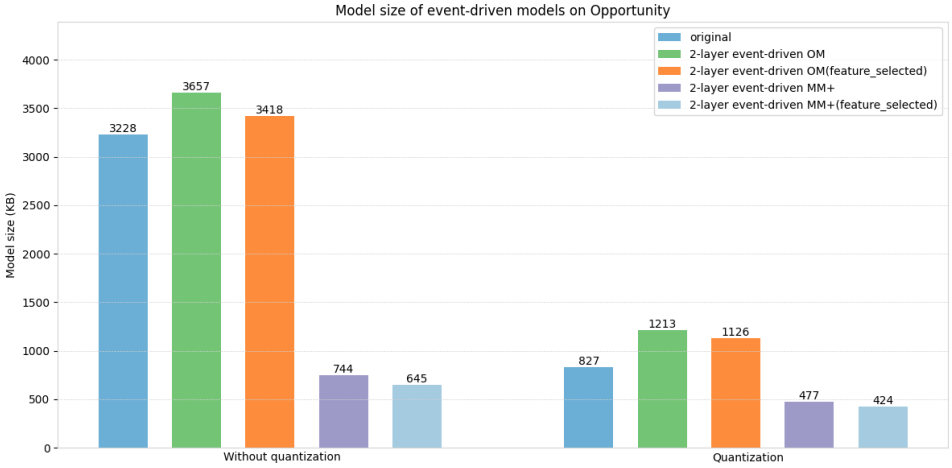

FIGURE 5.8: Model size of event-driven model on Opportunity

Similar to the Wisdm dataset, the event-driven model experienced a 45.7% increase in size as shown in Figure 5.8. Although feature selection slightly reduced the overall model size, it still remained larger than the OM. This highlights a key challenge in balancing the benefits of event-driven inference with the need to maintain a compact model size for resource-constrained applications. Through using optimal MM+ can largely reduce the model size, it still does not solve the problem as the extra space is needed for the pre-classification model.

To further enhance the potential of event-driven inference, the refinement of model paths looks like a promising solution. In addition to the existing simple and complex paths, introducing a medium path in between could provide significant benefits, especially when working with large and complex datasets that have numerous classes, such as Pamap2. To explore the feasibility of adding this additional medium path between the simple and complex paths, we applied the Event-driven Inference approach to the Pamap2 dataset. The Pamap2 dataset, with its high sampling rate of 100 Hz, which is three times higher than Opportunity and five times higher than Wisdm, and 27 diverse features, poses significant challenges for model. These characteristics result in longer inference times and larger model sizes, making it an ideal candidate for testing a more specific event-driven inference framework with three paths. In Pamap2, simple classes including lying, sitting, walking, running, cycling, nordic walking, ironing. While standing, ascending stairs, vac-

uum cleaning are medium classes. And rope jumping, vacuum cleaning are considered as complex classes. Feature selection is not used this time, as most of the features in Pamap2 are relatively important for precise prediction.

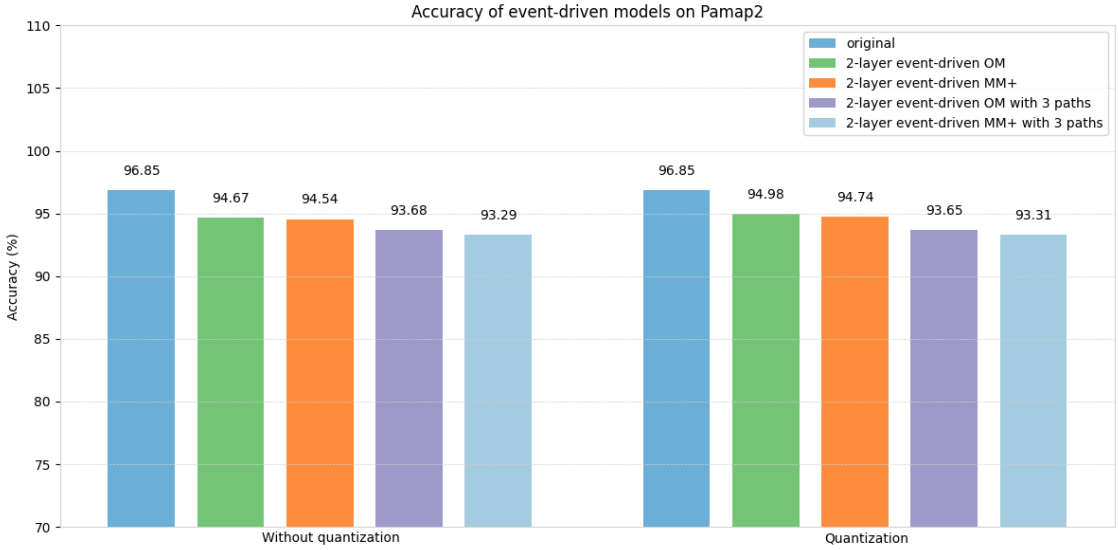

FIGURE 5.9: Accuracy of event-driven model on Pamap2

As shown in Figure 5.9, event-driven inference maintains a slight reduction in accuracy. The most significant drop is 3.56%, observed in the three-path event-driven model with optimal MM+. In this combination, the LM handles the simple path, the MM is assigned to the medium path, and the optimized MM+ is used for the complex path. For pre-classification accuracy, the two-path event-driven model achieves 96.98%, while the three-path model also maintains a high accuracy of 96.38%.

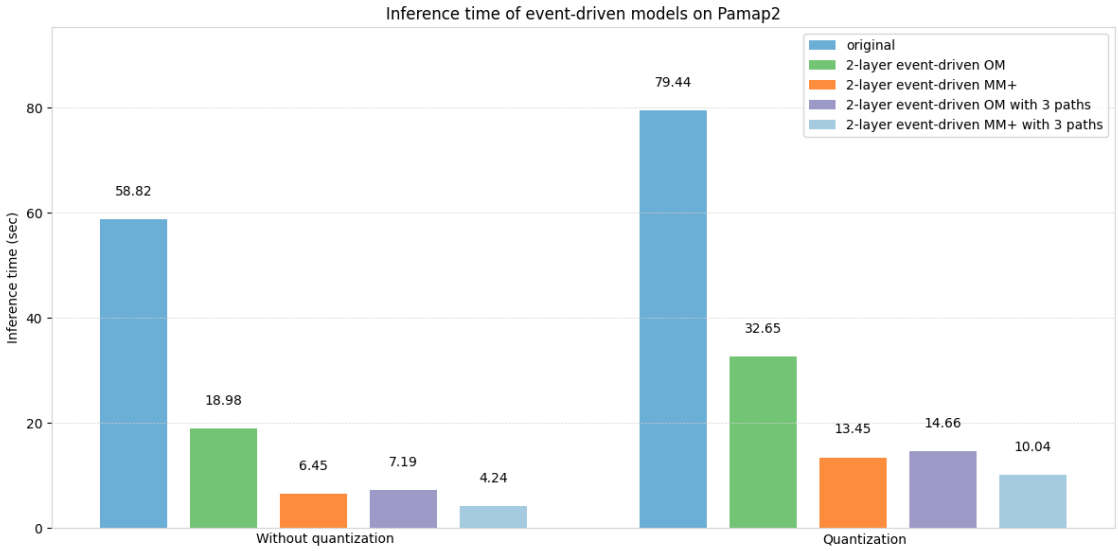

FIGURE 5.10: Inference time of event-driven model on Pamap2

Figure 5.10 demonstrates that both 2-path and 3-path event-driven inference methods significantly reduce inference time. Before applying quantization, the standard event-

driven model requires only 32% of the original model's inference time, while the event-driven model using optimal MM+ further reduces this to just 11%. Additionally, the 3-path event-driven model achieves remarkable efficiency, with inference times reduced to 12% and 7.2% of the original model's time.

For the quantized model, event-driven inference using the OM and optimized MM+ models significantly improves efficiency. Specifically, inference time is reduced to 41% and 17% of the quantized OM, respectively. When applying the 3-path event-driven approach, inference times are further reduced to 18.5% and 12.6%, depending on the chosen model. These results highlight the effectiveness of both 2-path and 3-path event-driven inference in achieving a balance between maintaining high accuracy and improving efficiency, particularly for complex datasets. Meanwhile, we again observe the increasing of inference time of tflite model in Figure 5.10 due to the reason we explain in the previous testing with Opportunity. This trend is also very obvious since Pamap2 is a quite large dataset because of the combination of high sampling rate and feature numbers.

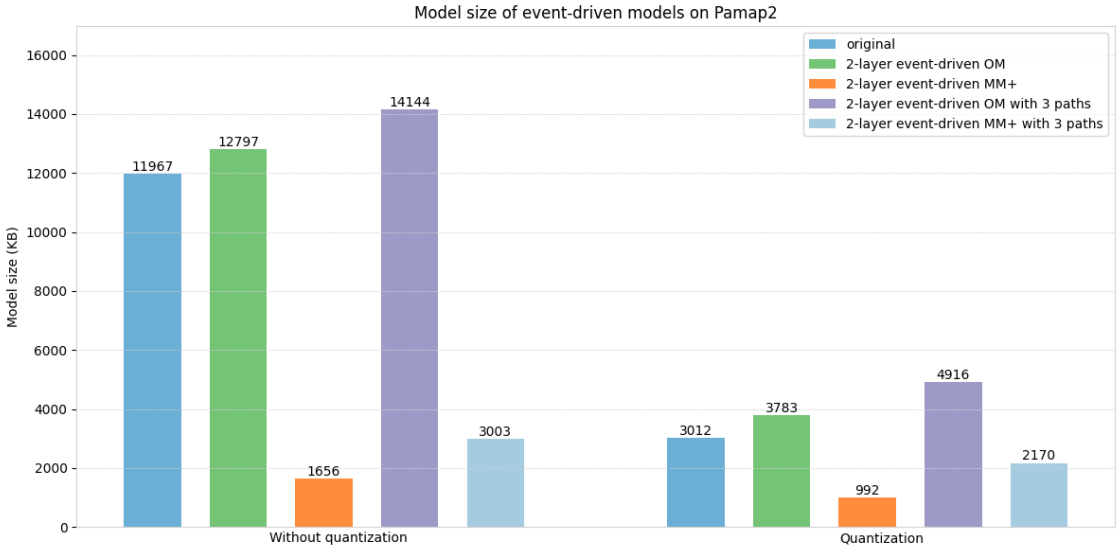

FIGURE 5.11: Model size of event-driven model on Pamap2

As illustrated in Figure 5.11, the model size increases due to the additional model required for event-driven inference. Replacing the OM with the optimized MM+ significantly reduces the model size, making it more suitable for edge devices, though challenges remain for constrained environments. For the 3-path event-driven inference, though it further reduces inference time, the overall model size grows due to the inclusion of an additional MM model for the medium path and an enlarged pre-classification model. To solve this challenge, the integrated event-driven model is introduced in Chapter 6.

We decided not to use simple and complex event-driven inference on the Sensors dataset because all model sizes perform very well on it. The accuracy difference between the LM and the OM is minimal, making it more efficient to use only the LM to save inference time. However, we applied momentary and durative event-driven inference to the Sensors dataset in Chapter 6 to explore the potential benefits of dividing activities into these two categories.

The findings of this section underscore the efficacy of event-driven inference in achieving substantial computational efficiency gains, particularly when combined with model quantization and feature selection. And in the case with complex datasets, an extra medium path

can further refine the pre-classification and further squeeze the inference time. The event-driven approach is especially beneficial for deployment in resource-constrained environments where inference speed and real-time capability are critical considerations. However, the increased model size introduced by the additional components in event-driven inference is a significant drawback that must be addressed to optimize the system's performance for practical use on resource-constrained devices, which will be discussed in Chapter 6.

Chapter 6

Integration of the "Event-driven inference" into a single model

In this chapter, we present the integration of the event-driven approach discussed in Chapter 5 into a unified model, aimed at simplifying deployment, enhancing efficiency, and maintaining robust performance. By consolidating the 2-layer event-driven model into a single, cohesive architecture, we eliminate the overhead associated with switching between multiple models, thereby reducing inference time and simplifying the deployment process by avoiding the need to manage multiple models.

The new integrated event-driven models are more than 30% faster than the previous 2-layer models. They only exhibit 7.85% to 35% of the inference time comparing with the OMs. Meanwhile, the attention mechanism is also applied on the integrated model, to further increase the accuracy to a close or even higher level of OM's with minimum cost of inference time and model size.

The concept of event-driven inference introduces dynamic routing, which is not fully supported by TensorFlow Lite for TFLM. Consequently, this technique cannot be deployed on devices running TFLM, such as the Arduino used in this research. Despite this limitation, the approach demonstrates its potential by significantly reducing inference time and improving accuracy when tested on the Raspberry Pi 5. These results highlight the feasibility of using event-driven inference in edge computing environments without using TFLM.

6.1 Methodology

The event-driven inference approach proposed in Chapter 5 theoretically reduces inference time by utilizing a simpler model when high complexity is unnecessary. However, it also introduces several challenges inherent to its two-layer structure, which involves the use of three distinct models. One such challenge is the overhead caused during transitions between models, which must be taken into account. Additionally, the output labels need adjustment to ensure consistency across different stages of inference. Furthermore, deploying three models with varying architectures on an edge device can complicate implementation and increase space requirements.

To address these issues, an integrated structure is proposed as presented in Figure 6.1. This integrated model is totally based on the DeepConvLSTM architecture, maintaining a similar design while retaining the benefits of event-driven inference. The key concept remains the same: to pre-classify samples into two categories. Based on the category, different layers within the model may be called and activated, thereby optimizing inference

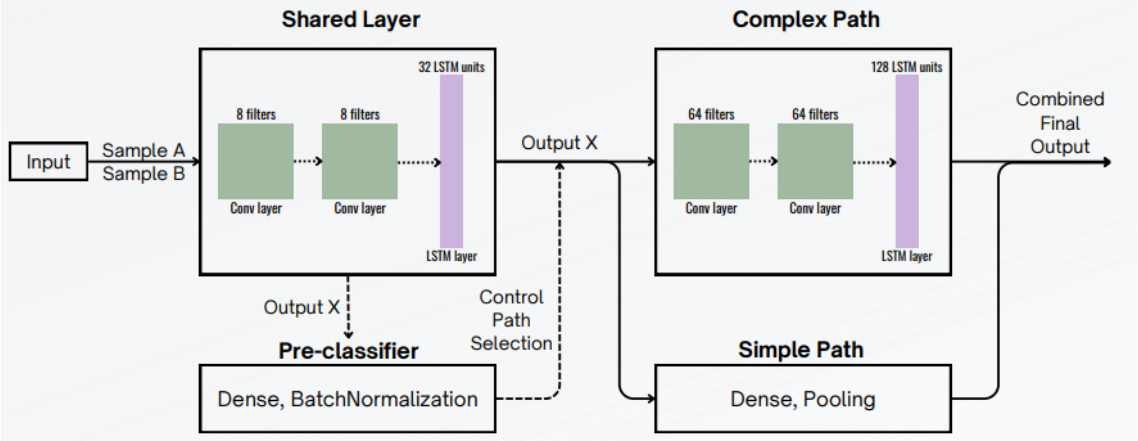

FIGURE 6.1: Structure design of integrated event-driven model

time while preserving accuracy. This approach aims to streamline deployment by reducing model complexity and improving resource efficiency on edge devices.

In this research, the concept of a shared layer is introduced to maximize computational efficiency. Initially, all samples are passed through the shared layers, which consist of small convolutional layers and LSTM layers that extract the preliminary features of the samples. Based on the output from the shared layers, the pre-classification layer, which includes its own hidden layer, normalization layer, and dropout layers, categorizes the samples as either simple or complex classes.

If a sample is classified as belonging to the simple class, it proceeds through the simple path. This simple path adds extra hidden layers on top of the output from the shared layer. If the sample is classified as belonging to the complex class, it follows the complex path, which applies additional convolutional layers and LSTM layers with the same number of filters and LSTM units in the OM. The important idea is both the simple and complex paths benefit from the computations performed by the shared layers, thereby minimizing redundant calculations. Each sample goes through the shared layer and pre-classification layer before selecting either the simple or complex path as shown in Figure 6.2.

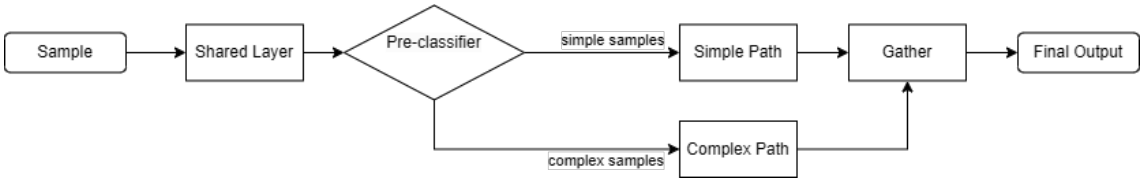

FIGURE 6.2: Flow of integrated event-driven model

Based on the result in Chapter 4, attention mechanisms and quantization were applied to the integrated event-driven model to enhance its efficiency and performance. Quantization proved effective in reducing both the model size and inference time, making it more suitable for deployment on resource-constrained devices. Additionally, the integration of attention mechanisms further improved the model's accuracy. However, knowledge distillation was not employed in this case due to the structural characteristics of the integrated model. Specifically, as the model's outputs are obtained from either the simple or complex path, it would be challenging for the teacher model to guide both paths in a right way.

6.2 Validation and Application

To evaluate the integrated event-driven model, we deployed it on the Raspberry Pi 5 for further testing. We compared the accuracy and inference time of this approach against both the original model and the 2-layer event-driven model to assess its effectiveness and improvements over the event-driven inference introduced in Chapter 5.

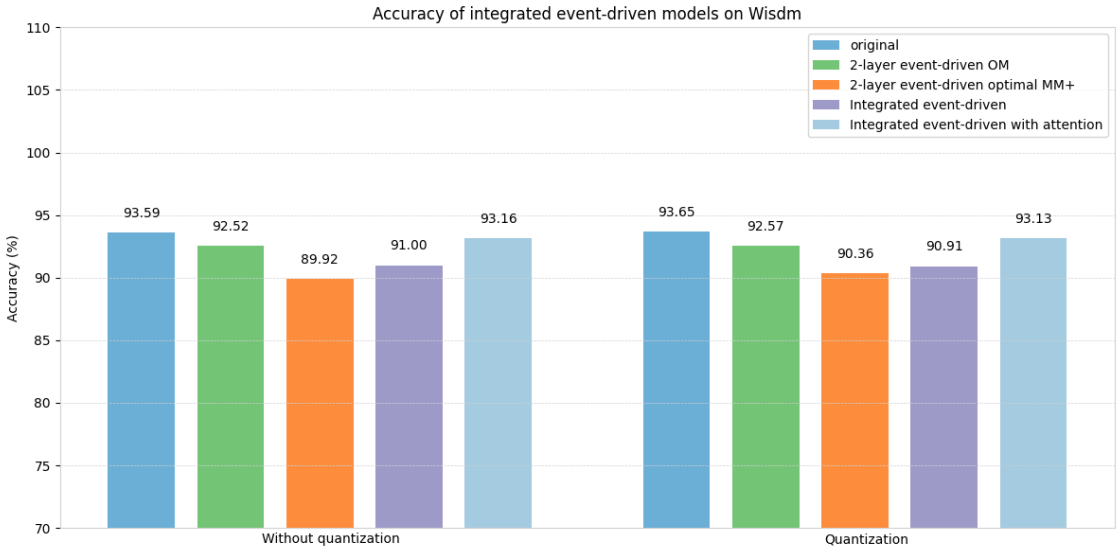

FIGURE 6.3: Accuracy of integrated event-driven model on Wisdm

The initial testing was conducted using the Wisdm dataset. As shown in Figure 6.3, the integrated event-driven model achieved accuracy levels comparable to the original model, similar to the performance of the two-layer event-driven model. The slight accuracy drop observed was within an acceptable range and was mainly caused by occasional misclassification during pre-classification, where complex samples were mistakenly processed by the simplified path. To avoid compromising efficiency by adding excessive layers to the pre-classification component, we decided to maintain its accuracy at approximately 93%, considering this minor trade-off reasonable. For the integrated model with attention mechanisms, the accuracy closely approached that of the OM, with the pre-classification accuracy further improving to 96.77%. This demonstrates the potential of attention mechanisms to enhance the overall performance of the event-driven model while maintaining computational efficiency.

More importantly, as shown in Figure 6.4, the integrated event-driven model reduced inference time to 34% of the OM's, which is also 30.65% faster than the two-layer event-driven model. This improvement is attributed by the elimination of the overhead involved in switching among three separate models and the effective reuse of outputs from shared layers. Additionally, the inclusion of attention introduced only a minimal increase in inference time, demonstrating its suitability for mitigating accuracy loss without significantly impacting computational efficiency.

Similarly, the integrated event-driven model demonstrated a reduction in inference time with post-quantization, as illustrated in Figure 6.4. In the case of dynamic range quantization, the integrated model reduced inference time to only 22.41% of that of the quantized OM. The inference times for the quantized 2-layer event-driven model and the integrated event-driven model were comparable, with the latter achieving approximately 23.53% acceleration. Even with the attention, the increase of inference time remains tiny.

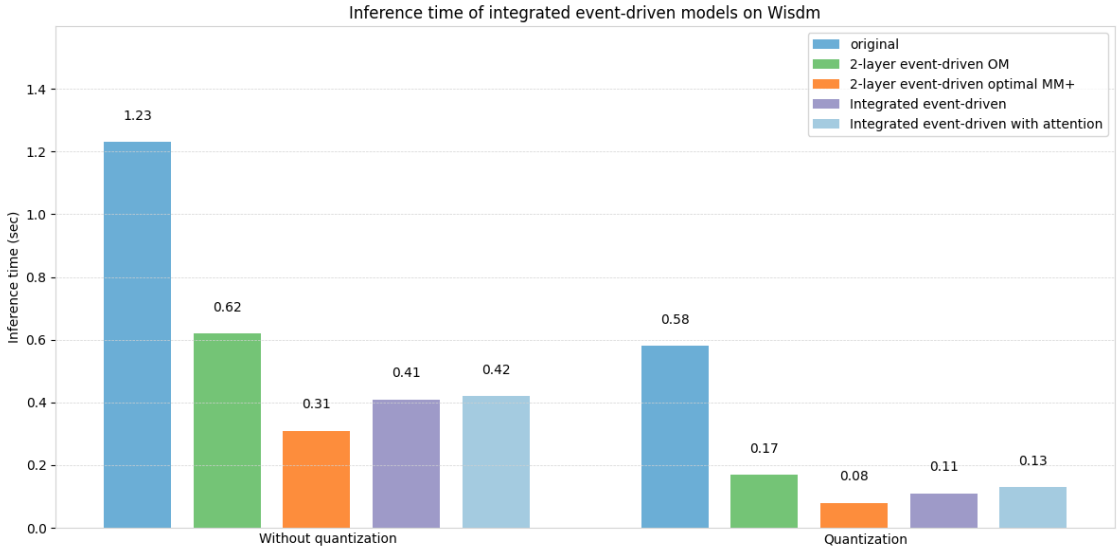

FIGURE 6.4: Inference time of integrated event-driven model on Wisdm

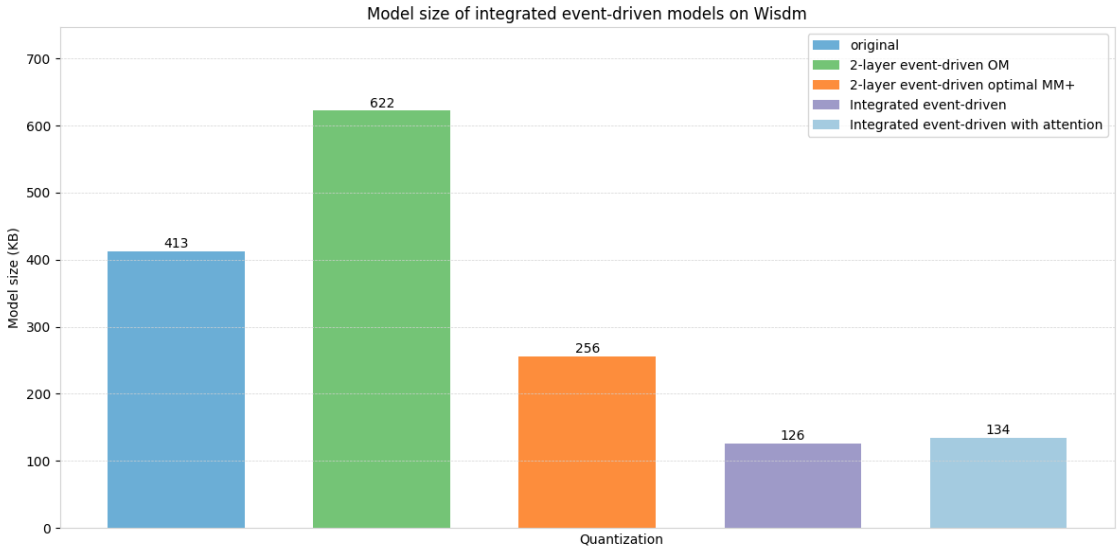

FIGURE 6.5: Model size of integrated event-driven model on Wisdm

In terms of model size, the integrated event-driven model effectively addresses the issue of increased size, as illustrated in Figure 6.5. The comparison focuses solely on the quantized TFLite version since the integrated event-driven model cannot be saved in the .h5 format. Remarkably, the integrated model is only one-third the size of the OM, making it highly suitable for deployment on smaller devices than a Raspberry Pi. Furthermore, the integrated model with attention is approximately 6% larger than the standard integrated model. This slight increase demonstrates that attention also only has a negligible impact on model size, ensuring its practicality for use in resource-constrained edge devices.

During testing with the Wisdm dataset, the integrated event-driven model with attention was identified as the best-performing model. It achieved an accuracy almost the same to the OM while requiring only 34% of the inference time and utilizing just 32.4% of the model size.

The new integrated model was further evaluated on the Opportunity dataset with feature selection applied. As illustrated in Figure 6.6, the accuracy of the new models is compared to the OM. Notably, four additional bars were added to the figure, representing the integrated event-driven models with and without attention and feature selection. Remarkably, these models even outperformed the OM in accuracy.

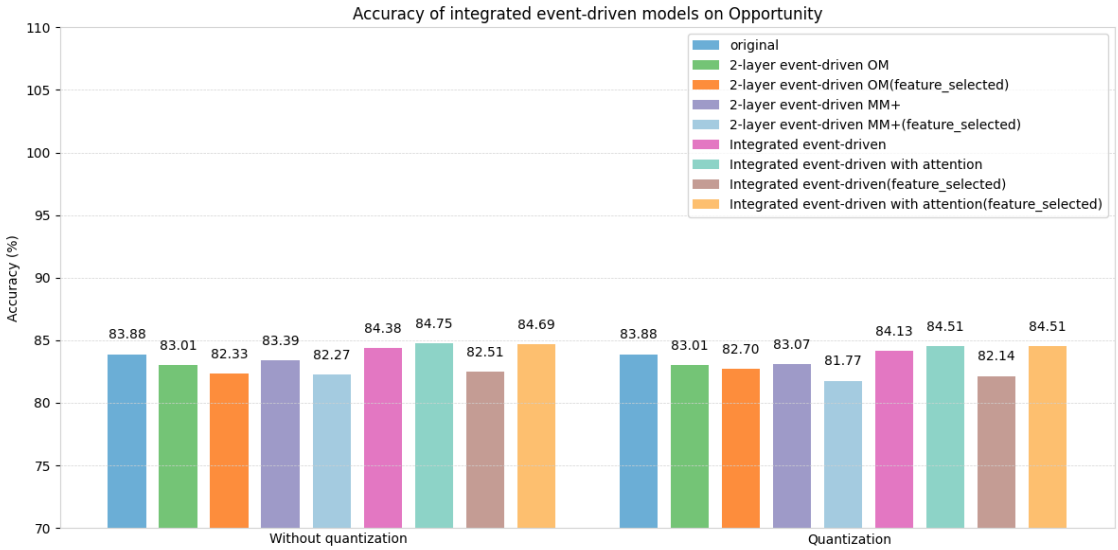

FIGURE 6.6: Accuracy of integrated event-driven model on Opportunity

This improvement can be attributed by the model design. During the training phase, the simple path is exclusively trained on simple samples, enabling it to specialize in distinguishing between simple activities. Similarly, the complex path is trained with complex samples, refining its ability in those scenarios. This specialization introduces a form of "domain-specific bias," where each path becomes more reliable within its respective domain. Provided the pre-classification accuracy remains high, which is 98.57% in this case, the simple and complex paths can rely on the pre-classifier's output rather than attempting to predict samples outside their domain. This approach enhances overall accuracy by leveraging the strengths of specialized paths.

When feature selection is also applied, attention mechanisms further enhance accuracy, bringing it very close to the results achieved with the full dataset. A similar trend is observed in the quantized models. The integrated model without attention, the integrated model with attention, and the integrated model with attention and feature selection all demonstrate superior performance compared to the OM in both accuracy and F1 score, underscoring the effectiveness of this approach.

When comparing inference time, the integrated models demonstrate excellent performance, as illustrated in Figure 6.7. All integrated models show lower inference times than the 2-layer event-driven model using the OM, and some are even faster than the one with MM+, while maintaining higher accuracy. Among these, the fastest model is the integrated model with attention and feature selection, achieving an inference time of only 13.3% of the OM's, while delivering superior accuracy. Compared to the fastest model identified in Chapter 5, which was the 2-layer event-driven model using the optimized MM+ with feature selection, the integrated model achieves slightly faster inference. Additionally, it improves accuracy by 2.42% and simplifies deployment by requiring only a single model. This makes the integrated approach a more practical and efficient solution for edge device

applications. In the case of quantized model, this trend is also maintained.

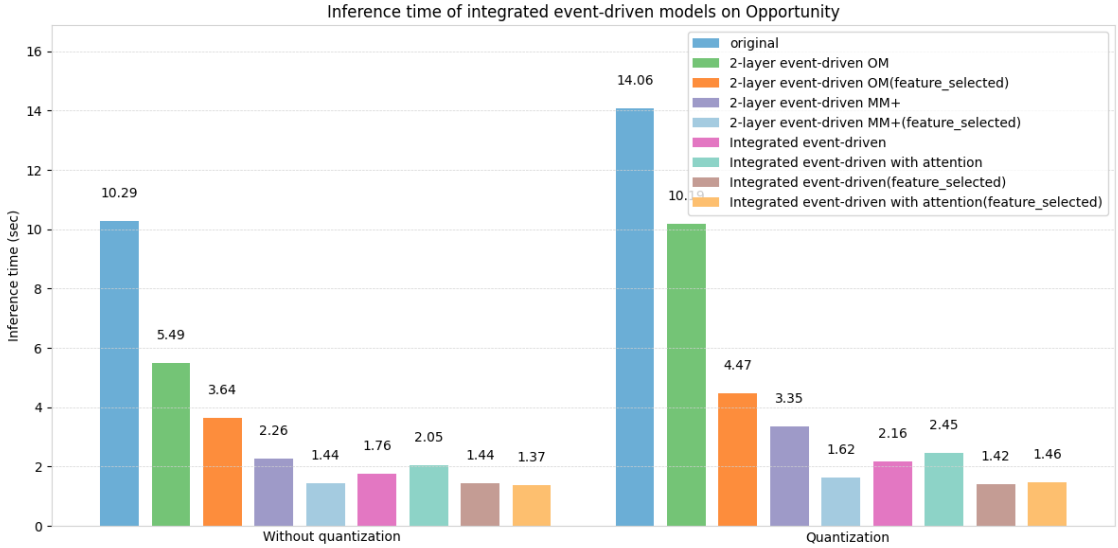

FIGURE 6.7: Inference time of integrated event-driven model on Opportunity

The integrated models address the concerns raised by the 2-layer models regarding model size. In Figure 6.8, depending on whether feature selection is applied, the integrated models are approximately 25% to 35% smaller than the OM. Compared to the 2-layer models introduced in Chapter 5, the integrated models are around 50% smaller. However, they remain slightly larger than the 2-layer models using the optimized MM+ because the integrated models retain the same number of filters and LSTM units in the complex path as the OM. While reducing these numbers could achieve a size similar to or smaller than the optimized MM+, this approach is avoided in this research to preserve the quantifiable improvements of integrated event-driven inference.

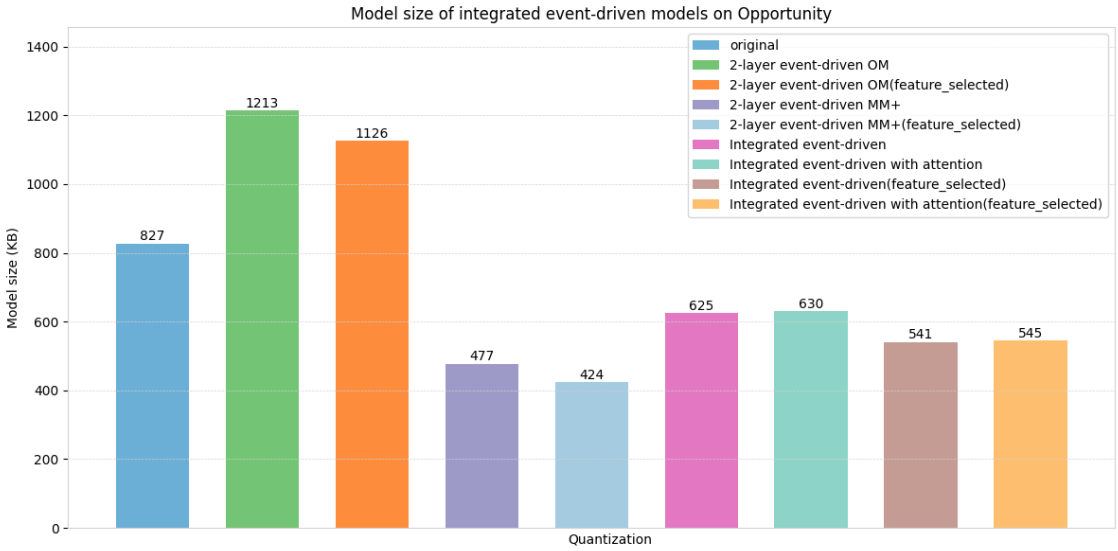

FIGURE 6.8: Model size of integrated event-driven model on Opportunity

It is also worth noting that the inclusion of attention mechanisms does not significantly

increase the model size. This ensures that any negative impact on the model remains minimal, making attention mechanisms a valuable technique for enhancing performance while maintaining a lightweight model structure.

During testing with the Opportunity dataset, the integrated event-driven model with attention and feature selection emerged as the best model. This model achieved even higher accuracy than the OM. The design of shared layers and specialized expert paths significantly reduced its inference time to just 13.3% of the OM's, while its model size was efficiently reduced to 65.9% of the original.

The final evaluation of the integrated event-driven model was conducted using the Pamap2 dataset. To enhance sample classification, an additional medium path was introduced again, similar to Section 5.2. As illustrated in Figure 6.9, all four integrated models demonstrated improved accuracy compared to the 2-layer models, achieving results that closely matched the performance of the OM. Notably, before quantization, the integrated model with attention achieved the highest accuracy. After quantization, the 3-path integrated model with attention outperformed the others, reaching the highest accuracy.

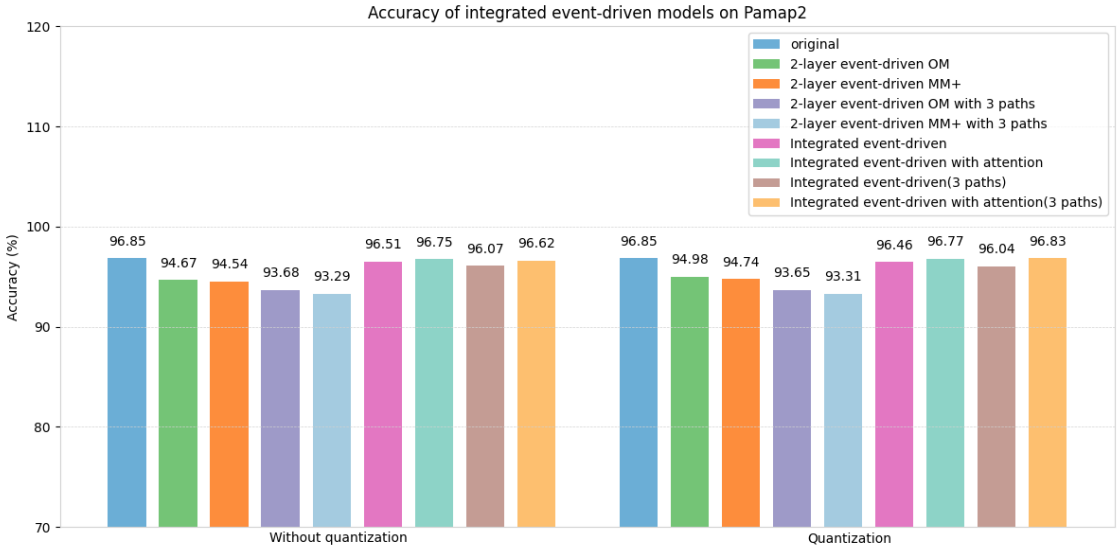

FIGURE 6.9: Accuracy of integrated event-driven model on Pamap2

For the Pamap2 dataset, the application of attention mechanisms continued to enhance accuracy. However, the observed improvements were less significant compared to results obtained with the Wisdm and Opportunity datasets. This outcome highlights that while attention mechanisms remain effective, their impact may vary depending on dataset characteristics and complexity.

The inference time results shown in Figure 6.10 highlight the efficiency of the new integrated models. Before quantization, the inference time is reduced to between 7.85% and 13.04% of the OM's time, representing a significant improvement. These models are approximately 70% faster than the 2-layer event-driven models. However, the 3-path integrated model does not provide additional reductions in inference time, likely due to the higher computational cost associated with dynamic routing. A similar trend is observed after quantization, where the new models remain approximately 70.9% to 77.1% faster than the OM, demonstrating their continued efficiency even in resource-constrained environments.

Despite this limitation, the evaluation confirms the overall effectiveness of the integrated

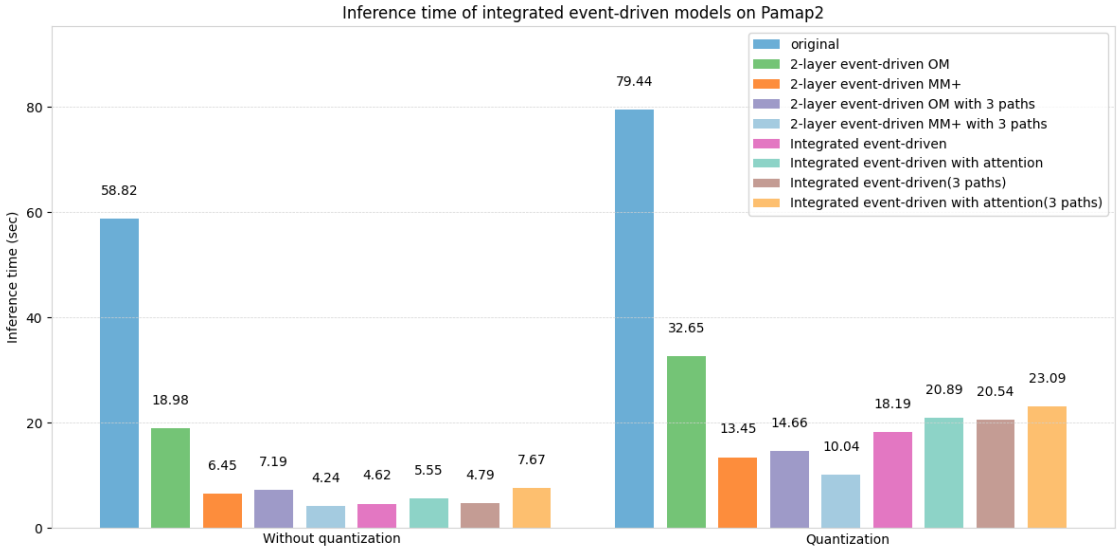

FIGURE 6.10: Inference time of integrated event-driven model on Pamap2

event-driven inference approach. It achieves substantial reductions in inference time while maintaining minimal impact on accuracy. Additionally, it resolves deployment challenges encountered with the 2-layer event-driven inference framework, making it a more practical solution for real-world applications.

The integrated models demonstrate a significant reduction in size as shown in Figure 6.11, achieving approximately 12% of the OM size. This outcome highlights the effectiveness of the integrated event-driven approach in addressing the challenges posed by the 2-layer event-driven inference. It reaches an optimal balance between accuracy, inference time, and model size, making it a more practical solution for real-world applications. Additionally, the inclusion of attention mechanism further enhances accuracy with only a minimal increase in model size, reinforcing its value in optimizing performance for resource-constrained environments.

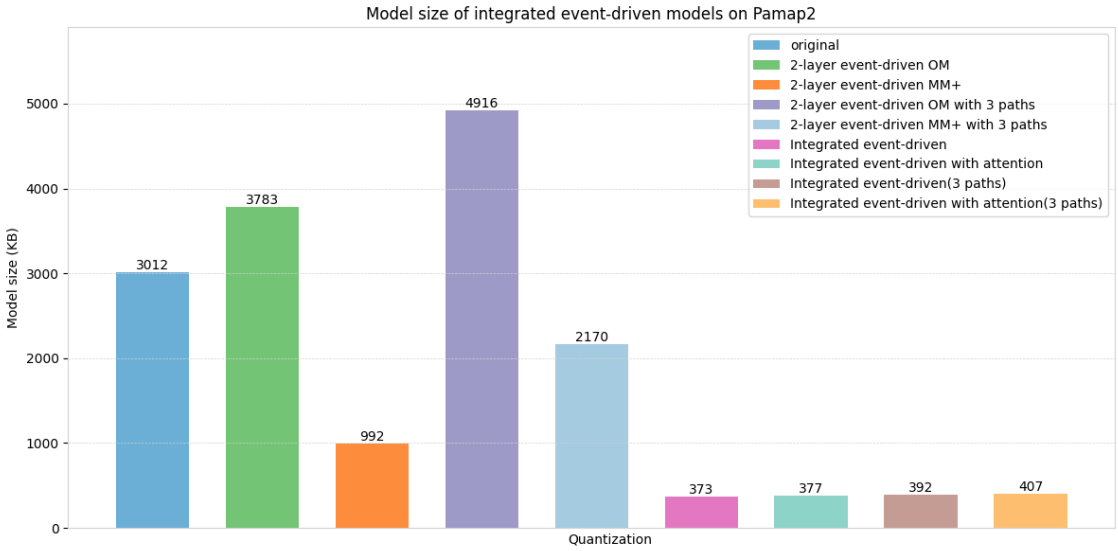

FIGURE 6.11: Model size of integrated event-driven model on Pamap2

During the testing with the Pamap2 dataset, the integrated event-driven model was identified as the most efficient model. It achieved very close accuracy levels to the OM, while significantly reducing inference time to just 7.85% of the OM's and minimizing the model size to 12.38% of the original.

For the Sensors dataset, we implement the durative and momentary event-driven inference method as outlined in Section 5.1. This approach is designed to replicate real-world scenarios where short-term actions occur within longer-duration activities. The dataset consists of seven activity classes: walking, standing, jogging, sitting, biking, going upstairs, and going downstairs. In this framework, standing is categorized as the momentary action, representing instances where the user temporarily pauses and stands still. The remaining activities are classified as durative actions, which persist for varying durations before being interrupted by the standing class or transitioning to another durative activity. Although the testing is performed on an existing dataset, this method effectively simulates real-life scenarios, providing a realistic evaluation of the event-driven inference approach.

In the testing, wee define the model structure similar to the model with simple and complex pre-classification. The sample first go through the shared layers with 2 convlutional layers and one lstm layer. Then they are pre-classified. The output includes the starting sample number and ending sample number of the period of durative samples, as well as the sample number of the momentary class. As we only has one momentary class, those samples are directly classified as standing. For the rest of durative samples, the middle samples of each period is sent to the durative path with extra 2 convolutional layers and one lstm layer for precise classification among 6 durative classes. Their output is then marked on all the other samples in this period.

During testing, we structured the model similarly to the one used for simple and complex event-driven inference. Each sample first passes through shared layers consisting of two convolutional layers and one LSTM layer. Following this, the samples are pre-classified. The output includes the starting and ending sample numbers for periods of durative activities, as well as the sample numbers for the momentary class. Since there is only one momentary class, these samples are directly classified as "standing." For the remaining durative samples, the samples in the middle of each period are sent to the durative path. This path includes additional two convolutional layers and one LSTM layer, which provide precise classification among the six durative classes. The resulting classification of the sample in the middle is then applied to all other samples within the same period, ensuring a fast inference with accurate labeling throughout.

As shown in Figure 6.12, the event-driven model, the event-driven model with attention, and the event-driven model with attention and feature selection all achieve a similarly high level of accuracy, which is comparable to that of the OM.

As shown in Figure 6.13, the momentary and durative event-driven inference method significantly reduces inference time to just 20% of the OM. Among the tested models, the integrated event-driven model with attention and feature selection performs the best, requiring only 15.5% of the OM's inference time while maintaining high accuracy.

A similar trend is observed in the quantized models. The integrated event-driven model reduces inference time to 45.9% of the OM, and the addition of attention and feature selection further decreases it to 28.1%. Notably, the CBAM again only has a minimal negative impact on inference time while significantly enhancing the accuracy of the model with feature selection, demonstrating its effectiveness in optimizing performance.

The testing conducted in this chapter demonstrates that integrated event-driven inference is a highly effective approach for reducing inference time in sensor-based HAR. Moreover, the integrated model addresses the challenges encountered in the 2-layer models

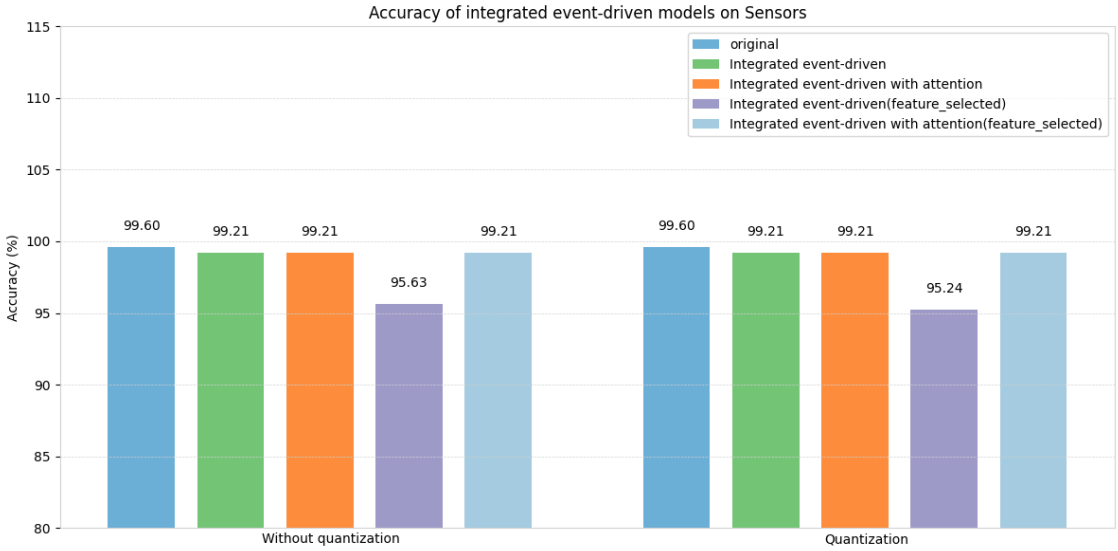

FIGURE 6.12: Accuracy of integrated event-driven model on Sensors

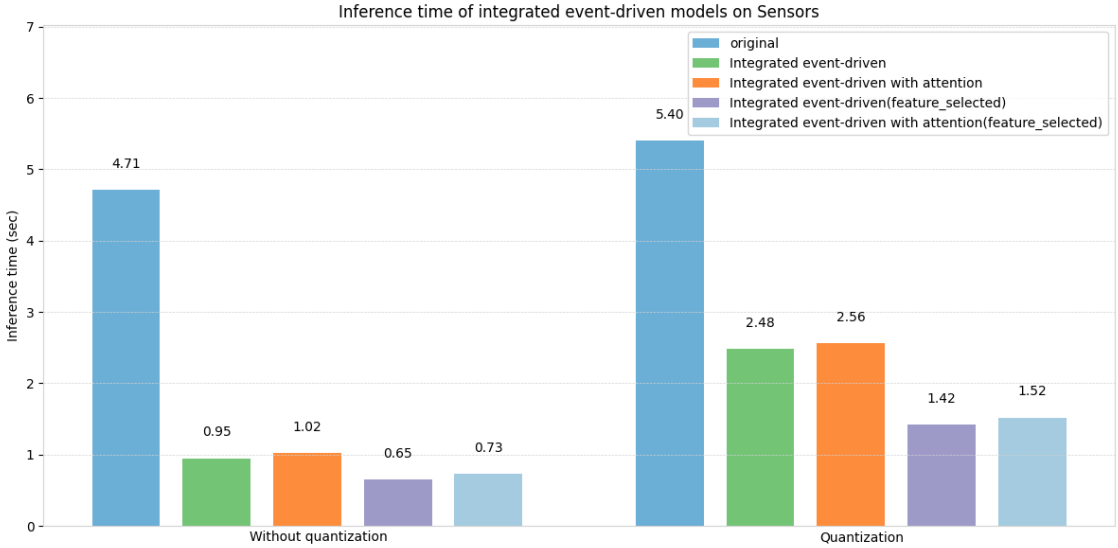

FIGURE 6.13: Inference time of integrated event-driven model on Sensors

discussed in Chapter 5, simplifying deployment while further decreasing both inference time and model size.

We evaluated event-driven inference using two classifications: simple and complex, as well as momentary and durative activities in this chapter. Both methods proved to be feasible and effective in enhancing efficiency. The integrated event-driven models require less than 35% of the inference time of the OM and are over 30% faster than the 2-layer event-driven model introduced in Chapter 5. These results underscore the potential of integrated event-driven inference as a practical solution for optimizing performance in resource-constrained environments.

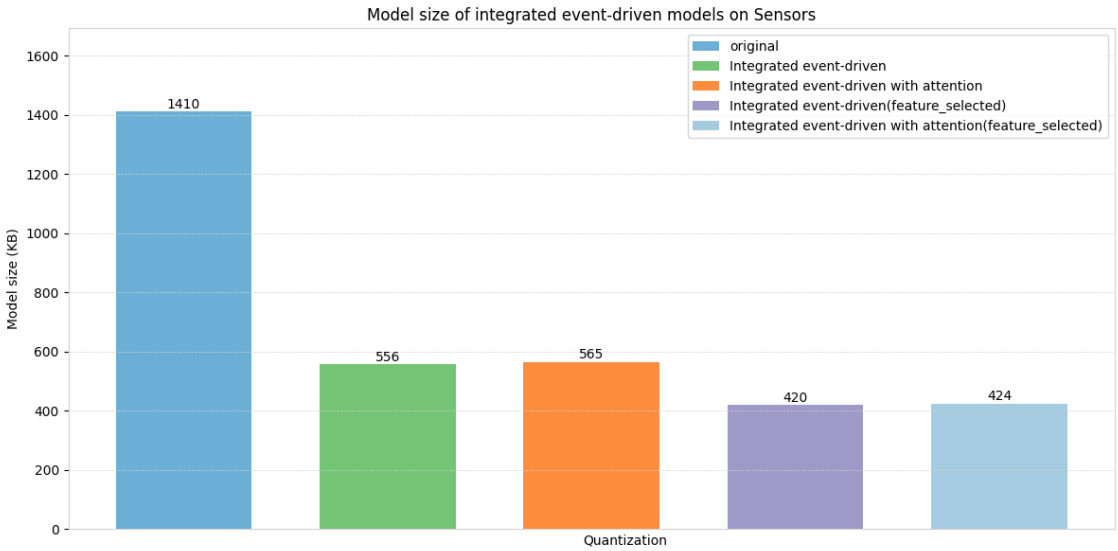

FIGURE 6.14: Model size of integrated event-driven model on Sensors

Chapter 7

Conclusion

7.1 Integration

In this thesis, we explored various techniques to optimize a complex sensor-based HAR model for deployment on edge devices. The primary goal was to reduce the model size and inference time while maintaining high accuracy and addressing compatibility challenges.

To achieve this, we began by evaluating existing optimization techniques such as quantization, pruning, knowledge distillation, and attention mechanisms in Chapters 2 and 3. After assessing their effectiveness, we selected the most promising techniques and combined them into a single model in Chapter 4. This optimized model achieved accuracy levels comparable to the OM while reducing the model size to just 3% of the OM's size. These optimized models were successfully deployed on the Arduino Nano 33 BLE Sense, demonstrating the feasibility of running a DeepConvLSTM model enhanced with attention and knowledge distillation on a highly resource-constrained device.

Building on this foundation, we analyzed the characteristics of sensor-based HAR and introduced the concept of event-driven inference to further improve inference time. In Chapter 5, we implemented this concept using three separate models to test its feasibility and effectiveness. This approach reduced inference time by more than half, confirming the value of event-driven inference. However, it also introduced new challenges related to deployment and increased model size due to the additional models required.

To address these issues, we developed integrated event-driven models in Chapter 6. These models incorporate event-driven inference within a single structure by modifying the architecture of DeepConvLSTM. By applying attention mechanisms to these models, we achieved an even more accurate system than the OM, with inference times reduced to less than 35% of the OM's. This performance was also over 30% faster than the 2-layer event-driven model proposed in Chapter 5.

Although the dynamic routing capabilities of these models prevent their deployment on the Arduino Nano due to limitations in TFLM, they achieved significant efficiency improvements on the Raspberry Pi 5. These results highlight the potential of event-driven inference to enhance efficiency in sensor-based HAR, especially on slightly more capable edge devices.

7.2 Contribution

This thesis effectively addresses the four research gaps identified in Chapter 1. For the ***first research gap***, we identified the primary cause of the accuracy drop in full integer quantization and mitigated it through the application of feature selection. In relation to the

second research gap and ***third research gap***, we thoroughly evaluated the performance of attention mechanisms and successfully deployed the DeepConvLSTM model enhanced with knowledge distillation and attention on edge devices. Lastly, for the ***fourth research gap***, we introduced event-driven inference to address the challenges discussed. Drawing inspiration from NAS and DRN, the integrated event-driven inference approach successfully overcomes their challenges and delivers strong overall performance.

7.3 Future Work

While significant improvements in efficiency were achieved through existing techniques and the newly proposed event-driven inference, several unresolved issues and opportunities for future research remain. Although we successfully deployed the compact models on the Arduino Nano 33 BLE Sense, there is still a noticeable accuracy drop compared to the OMs. Additionally, we have yet to fully address the increased model size and inference time that result when the DeepConvLSTM model is quantized for TFLM and saved in the TFLite format. Future work could focus on developing methods to optimize LSTM layers for better compatibility with TFLM and the TFLite format.

Regarding event-driven inference, while it significantly reduces inference time, it cannot currently be deployed on devices running TFLM due to the lack of support for dynamic routing. This limitation is a major barrier preventing event-driven inference from functioning effectively on tiny edge devices. Further testing and development are needed to explore its compatibility with newer versions of TFLM. Additionally, this approach should be tested in real-time inference scenarios to validate its effectiveness in practical applications. Event-driven inference may also prove more effective in image-based HAR tasks, where a larger gap between the simple and complex paths could maximize its efficiency.

Moreover, a more comprehensive evaluation should be conducted on the durative and momentary event-driven inference, as this approach closely aligns with real-world applications involving the detection of complex and fast activities. Such detailed testing could provide further insights into its potential for practical deployment and real-life utility.

Appendix I

Keras Model name	Accuracy(%)	Model Size(KB)	Inference Time(s)
Opportunity			
OM	83.88	3277	10.29
MM+	82.89	309	2.54
MM	77.47	130	1.96
LM	74.05	73	1.26
MM+ KD Att	84.32	321	2.71
2-layer event-driven OM	83.01	—	5.49
2-layer event-driven OM feature selection	82.33	—	3.64
2-layer event-driven MM+	83.39	—	2.26
2-layer event-driven MM+ feature selection	82.27	—	1.44
Integrated event-driven	84.38	—	1.76
Integrated event-driven Att	84.75	—	2.05
Integrated event-driven feature selection	82.51	—	1.44
Integrated event-driven feature selection Att	84.69	—	1.37
Sensors			
OM	99.60	5562	4.71
MM+	97.62	412	1.22
MM	97.22	142	0.72
LM	97.22	69	0.57
MM+ KD Att	98.81	425	1.26
Integrated event-driven	99.21	—	0.95
Integrated event-driven Att	99.21	—	1.02
Integrated event-driven feature selection	95.63	—	0.65
Integrated event-driven feature selection Att	99.21	—	0.73
Wisdm			
OM	93.59	1573	1.23
MM+	91.03	136	0.68
MM	89.37	65	0.51
LM	83.40	45	0.42
MM+ KD Att	91.99	148	0.70
2-layer event-driven OM	92.52	—	0.62
2-layer event-driven MM+	89.92	—	0.61
Integrated event-driven	91.00	—	0.41
Integrated event-driven Att	93.16	—	0.42
Pamap2			
OM	96.85	11967	58.82
MM+	95.26	814	13.42
MM	93.31	243	8.13
LM	91.86	95	4.82
MM+ KD Att	96.41	826	14.26
2-layer event-driven OM	94.67	—	18.98
2-layer event-driven MM+	94.54	—	6.45
2-layer event-driven OM 3 path	93.68	—	7.19
2-layer event-driven MM+ 3 path	93.29	—	4.24
Integrated event-driven	96.51	—	4.62
Integrated event-driven Att	96.75	—	5.55
Integrated event-driven 3 path	96.07	—	4.79
Integrated event-driven 3 path Att	96.62	—	7.67

TABLE 1: Results of Keras Models

Appendix II

Quantized TFlite Model name	Accuracy(%)	Model Size(KB)	Inference Time(s)
Opportunity			
OM	83.88	827	14.06
MM+	82.95	250	3.12
MM	77.47	93	2.38
LM	73.99	32	1.52
MM+ KD Att	84.01	264	4.58
2-layer event-driven OM	83.01	1213	10.19
2-layer event-driven OM feature selection	82.70	1126	4.47
2-layer event-driven MM+	83.07	477	3.35
2-layer event-driven MM+ feature selection	81.77	424	1.62
Integrated event-driven	84.13	625	2.16
Integrated event-driven Att	84.51	630	2.45
Integrated event-driven feature selection	82.14	541	1.42
Integrated event-driven feature selection Att	84.51	545	1.46
Sensors			
OM	99.60	1410	5.4
MM+	97.62	119	1.19
MM	97.62	50	0.83
LM	97.22	31	0.52
MM+ KD Att	98.02	120	1.23
Integrated event-driven	99.21	556	2.48
Integrated event-driven Att	99.21	565	2.56
Integrated event-driven feature selection	95.24	420	1.42
Integrated event-driven feature selection Att	99.21	424	1.52
Wisdm			
OM	93.65	413	0.58
MM+	90.91	47	0.37
MM	89.34	29	0.31
LM	83.40	25	0.19
MM+ KD Att	92.22	53	0.42
2-layer event-driven OM	92.57	622	0.17
2-layer event-driven MM+	90.36	256	0.08
Integrated event-driven	90.91	126	0.11
Integrated event-driven Att	93.13	134	0.13
Pamap2			
OM	96.85	3012	79.44
MM+	95.26	219	28.26
MM	93.13	75	20.58
LM	92.01	36	13.21
MM+ KD Att	96.41	221	29.68
2-layer event-driven OM	94.98	3783	32.65
2-layer event-driven MM+	94.74	992	13.45
2-layer event-driven OM 3 path	93.65	4916	14.66
2-layer event-driven MM+ 3 path	93.31	2170	10.04
Integrated event-driven	96.46	373	18.19
Integrated event-driven Att	96.77	377	20.89
Integrated event-driven 3 path	96.07	392	20.54
Integrated event-driven 3 path Att	96.83	407	23.09

Table 2: Result of Quantized TFLite Models

Bibliography

[1] Edge Impulse - The Leading Edge AI Platform. URL: https://edgeimpulse.com/.

[2] Sumeyye Agac and Ozlem Durmaz Incel. Resource-efficient, sensor-based human activity recognition with lightweight deep models boosted with attention. *Computers and Electrical Engineering*, 117:109274, 2024. URL: https://www.sciencedirect.com/science/article/pii/S0045790624002027, doi:10.1016/j.compeleceng.2024.109274.

[3] Sevda Ozge Bursa, Ozlem Durmaz Incel, and Gulfem Isiklar Alptekin. Transforming deep learning models for resource-efficient activity recognition on mobile devices. In *2022 5th Conference on Cloud and Internet of Things (CIoT)*, pages 83–89, 2022. doi:10.1109/CIoT53061.2022.9766512.

[4] Shaofeng Cai, Yao Shu, and Wei Wang. Dynamic routing networks. In *proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 3588–3597, 2021.

[5] Ricardo Chavarriaga, Hesam Sagha, Alberto Calatroni, Sundara Tejaswi Digumarti, Gerhard Tröster, José del R. Millán, and Daniel Roggen. The Opportunity challenge: A benchmark database for on-body sensor-based activity recognition. *Pattern Recognition Letters*, 34(15):2033–2042, 2013. URL: https://www.sciencedirect.com/science/article/pii/S0167865512004205, doi:10.1016/j.patrec.2012.12.014.

[6] Chiara Contoli and Emanuele Lattanzi. A study on the application of tensorflow compression techniques to human activity recognition. *IEEE Access*, 11:48046–48058, 2023. doi:10.1109/ACCESS.2023.3276438.

[7] Francesco Daghero, Alessio Burrello, Chen Xie, Marco Castellano, Luca Gandolfi, Andrea Calimera, Enrico Macii, Massimo Poncino, and Daniele Jahier Pagliari. Human activity recognition on microcontrollers with quantized and adaptive deep neural networks. *ACM Trans. Embed. Comput. Syst.*, 21(4), August 2022. doi:10.1145/3542819.

[8] Shizhuo Deng, Jiaqi Chen, Da Teng, Chuangui Yang, Dongyue Chen, Tong Jia, and Hao Wang. Lhar: Lightweight human activity recognition on knowledge distillation. *IEEE Journal of Biomedical and Health Informatics*, 2023.

[9] Wenbin Gao, Lei Zhang, Qi Teng, Jun He, and Hao Wu. Danhar: Dual attention network for multimodal human activity recognition using wearable sensors. *Applied Soft Computing*, 111:107728, 2021.

[10] Alessandro Ghibellini, Luciano Bononi, and Marco Di Felice. Intelligence at the iot edge: Activity recognition with low-power microcontrollers and convolutional neural networks. In *2022 IEEE 19th Annual Consumer Communications Networking Conference (CCNC)*, pages 707–710, 2022. doi:10.1109/CCNC49033.2022.9700665.

[11] Zhenyu He, Yulin Sun, and Zhen Zhang. Human Activity Recognition Based on Deep Learning Regardless of Sensor Orientation. *Applied Sciences*, 14(9):3637, January 2024. Number: 9 Publisher: Multidisciplinary Digital Publishing Institute. URL: https://www.mdpi.com/2076-3417/14/9/3637, doi:10.3390/app14093637.

[12] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018.

[13] Malihe Mardanpour, Majid Sepahvand, Fardin Abdali-Mohammadi, Mahya Nikouei, and Homeyra Sarabi. Human activity recognition based on multiple inertial sensors through feature-based knowledge distillation paradigm. *Information Sciences*, 640:119073, 2023.

[14] Mason McGill and Pietro Perona. Deciding how to decide: Dynamic routing in artificial neural networks. In *International Conference on Machine Learning*, pages 2363–2372. PMLR, 2017.

[15] Pierre-Emmanuel Novac, Ghouthi Boukli Hacene, Alain Pegatoquet, Benoît Miramond, and Vincent Gripon. Quantization and Deployment of Deep Neural Networks on Microcontrollers. *Sensors*, 21(9):2984, January 2021. Number: 9 Publisher: Multidisciplinary Digital Publishing Institute. URL: https://www.mdpi.com/1424-8220/21/9/2984, doi:10.3390/s21092984.

[16] Francisco Javier Ordóñez and Daniel Roggen. Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition. *Sensors*, 16(1):115, January 2016. Number: 1 Publisher: Multidisciplinary Digital Publishing Institute. URL: https://www.mdpi.com/1424-8220/16/1/115, doi:10.3390/s16010115.

[17] KwapiszJennifer R, WeissGary M, and MooreSamuel A. Activity recognition using cell phone accelerometers. *ACM SIGKDD Explorations Newsletter*, March 2011. Publisher: ACMPUB27New York, NY, USA. URL: https://dl.acm.org/doi/10.1145/1964897.1964918, doi:10.1145/1964897.1964918.

[18] Attila Reiss and Didier Stricker. Introducing a New Benchmarked Dataset for Activity Monitoring. In *2012 16th International Symposium on Wearable Computers*, pages 108–109, 2012. doi:10.1109/ISWC.2012.13.

[19] Tao Sheng, Chen Feng, Shaojie Zhuo, Xiaopeng Zhang, Liang Shen, and Mickey Aleksic. A quantization-friendly separable convolution for mobilenets. In *2018 1st Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications (EMC2)*, pages 14–18. IEEE, 2018.

[20] Muhammad Shoaib, Stephan Bosch, Ozlem Durmaz Incel, Hans Scholten, and Paul J. M. Havinga. Fusion of Smartphone Motion Sensors for Physical Activity Recognition. *Sensors*, 14(6):10146–10176, June 2014. Number: 6 Publisher: Multidisciplinary Digital Publishing Institute. URL: https://www.mdpi.com/1424-8220/14/6/10146, doi:10.3390/s140610146.

[21] Muhammad Shoaib, Hans Scholten, Paul J. M. Havinga, and Ozlem Durmaz Incel. A hierarchical lazy smoking detection algorithm using smartwatch sensors. In *2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom)*, pages 1–6, 2016. doi:10.1109/HealthCom.2016.7749439.

[22] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 409–424, 2018.

[23] Hao Wu, Patrick Judd, Xiaojie Zhang, Mikhail Isaev, and Paulius Micikevicius. Integer Quantization for Deep Learning Inference: Principles and Empirical Evaluation, April 2020. arXiv:2004.09602 [cs]. URL: http://arxiv.org/abs/2004.09602, doi:10.48550/arXiv.2004.09602.

[24] Fuyuka Yamada, Satoki Tsuji, Hiroshi Kawaguchi, Atsuki Inoue, and Yasufumi Sakai. A high-speed neural architecture search considering the number of weights. In *KI 2021: Advances in Artificial Intelligence: 44th German Conference on AI, Virtual Event, September 27–October 1, 2021, Proceedings 44*, pages 109–115. Springer, 2021.

[25] Ming Zeng, Haoxiang Gao, Tong Yu, Ole J Mengshoel, Helge Langseth, Ian Lane, and Xiaobing Liu. Understanding and improving recurrent networks for human activity recognition by continuous attention. In *Proceedings of the 2018 ACM international symposium on wearable computers*, pages 56–63, 2018.

[26] B Zoph. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.