On the Design of Efficient Deep Learning Methods for Human Activity Recognition in Resource Constrained Devices

by

Sheikh Nooruddin

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2023

Author's Declaration

Statement of Contributions

Following publications have resulted from the work presented in the thesis:

1. **Sheikh Nooruddin**, Md. Milon Islam, Fakhri Karray and Ghulam Muhammad, "A Multi-resolution Fusion Approach for Human Activity Recognition from Video Data in Tiny Edge Devices," Information Fusion, Elsevier, 2023. [Under Review].

2. **Sheikh Nooruddin**, Md. Milon Islam and Fakhri Karray, "TinyHAR: Benchmarking Human Activity Recognition Systems in Resource Constrained Devices," 8^{th} IEEE World Forum on Internet of Things (IEEE WFIoT2022), IEEE, Yokohama, 26 Oct.–11 Nov., 2022. [Accepted].

3. **Sheikh Nooruddin**, Md. Milon Islam and Fakhri Karray, "An Evolutionary Computing based Approach for Optimal Target Coverage in Wireless Sensor Networks," Proceedings of 15^{th} KES International Conference on Human Centred Intelligent Systems (HCIS), edited by: Zimmermann, A., Howlett, R.J., Jain, L.C., Human Centred Intelligent Systems. Smart Innovation, Systems and Technologies, vol. 310, Springer, Singapore, 2022.

4. Md. Milon Islam, **Sheikh Nooruddin**, Fakhri Karray and Ghulam Muhammad, "Multi-level Feature Fusion for Multimodal Human Activity Recognition in Internet of Healthcare Things," Information Fusion, Elsevier, vol. 94, pp. 17-31, Jan. 2023.

5. Md. Milon Islam, **Sheikh Nooruddin**, Fakhri Karray and Ghulam Muhammad, "Internet of Things: Device Capabilities, Architectures, Protocols, and Smart Applications in Healthcare Domain," IEEE Internet of Things Journal, IEEE, vol. 10, no. 4, pp. 3611-3641, Dec. 2022.

6. Md. Milon Islam, **Sheikh Nooruddin**, Fakhri Karray and Ghulam Muhammad, "Human Activity Recognition Using Tools of Convolutional Neural Networks: A State of the Art Review, Data Sets, Challenges, and Future Prospects," Computers in Biology and Medicine, Elsevier, vol. 149, pp. 106060, Oct. 2022.

7. Md. Milon Islam, **Sheikh Nooruddin** and Fakhri Karray "Multimodal Human Activity Recognition for Smart Healthcare Applications," IEEE International Conference on Systems, Man, and Cybernetics (SMC), IEEE, Prague, pp. 196-203, 9-12 Oct., 2022.

In papers 1, 2, and 3, I was responsible for the conceptualization, methodology, validation, investigation, and writing the original draft. Md. Milon Islam was partly responsible for the methodology, validation, investigation, and writing the original draft.

In papers 4, 5, 6, and 7, I was responsible partly for the methodology, literature review, data presentation, discussion, and writing the original draft. Md. Milon Islam was responsible for the conceptualization, methodology, literature review, data presentation, discussion, and writing the original draft.

In all of the above papers, Prof. Fakhri Karray was responsible for the review, revision, coordination, guidance, and suggestions. Prof. Ghulam Muhammad aided in the review, revision, guidance, and suggestions.

Chapters 1 to 5 of this dissertation contain relevant parts of papers 1 to 7.

Abstract

Human Activity Recognition (HAR) is the process of automatic recognition of Activities of Daily Life (ADL) from human motion data captured in various data modalities by wearable and ambient sensors. Advances in deep learning, especially Convolutional Neural Networks (CNN) have revolutionized intelligent frameworks such as HAR systems by effectively and efficiently inferring human activity from various modalities of data. However, the training and inference of CNNs are often resource-intensive. Recent research developments are focused on bringing the effectiveness of CNNs in resource constrained edge devices through Tiny Machine Learning (TinyML). TinyML aims to optimize these models in terms of compute and memory requirements - aiming to make them suitable for always-on resource constrained devices - leading to a reduction in communication latency and network traffic for HAR frameworks. In this thesis, at first, we provide a benchmark to understand these trade-offs among variations of CNN network architectures, different training methodologies, and different modalities of data in the context of HAR, TinyML, and edge devices. We tested and reported the performance of CNN and Depthwise Separable Convolutional Neural Network (DSCNN) models as well as two training methodologies: Quantization Aware Training (QAT) and Post Training Quantization (PTQ) on five commonly used benchmark datasets containing image and time-series data: UP-Fall, Fall Detection Dataset (FDD), PAMAP2, UCI-HAR, and WISDM. We also deployed and tested the performance of the model-based standalone applications on multiple commonly available resource constrained edge devices in terms of inference time and power consumption. Later, we focus on HAR from video data sources. We proposed a two-stream multi-resolution fusion architecture for HAR from video data modality. The context stream takes a resized image as input and the fovea stream takes the cropped center portion of the resized image as input, reducing the overall dimensionality. Due to camera bias, objects of interest are often situated in the center of the frame. We tested two quantization methods: PTQ and QAT to optimize these models for deployment in edge devices and tested the performance in two challenging video datasets: KTH and UCF11. We performed ablation studies to validate the two-stream model performance. We deployed the proposed architecture in commercial resource constrained devices and monitored their performance in terms of inference latency and power consumption. The results indicate that the proposed architecture clearly outperforms other relevant single-stream models tested in this work in terms of accuracy, precision, recall, and F1 score while also reducing the overall model size. The experimental results in this thesis demonstrate the effectiveness and feasibility of TinyML for HAR from multimodal data sources in edge devices.

Acknowledgements

I would like to thank my supervisor Prof. Fakhri Karray for all the guidance and support given to me throughout my Master's degree as well as the research opportunities.

In addition, I want to thank my colleague Md. Milon Islam for our long brainstorming sessions at the CPAMI lab. I also want to thank Prof. Ghulam Muhammad for reviewing, revising, and suggesting meaningful changes to our manuscripts.

I would like to acknowledge and thank the members of my thesis committee Prof. Mark Crowley and Prof. Ali Elkamel for reviewing this thesis. Their assessment and evaluation are very appreciated.

I would love to express my deepest gratitude to my wife Falguni Ahmed Sharna, without whose selfless sacrifice, dedication, and unwavering support, none of this would have been possible.

Dedication

This thesis is dedicated to my wife, parents, colleagues, and those who helped me push through until the end.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ACGAN Auxiliary Classifier Generative Adversarial Network 15

ADL Activities of Daily Life 16, 33

AUC Area Under Curve 55

Bi-LSTM Bi-Directional Long Short-Term Memory 11, 13

BLE Bluetooh Low Energy 26

CNN Convolutional Neural Network 3, 5–8, 10–17, 19–25, 32, 38, 41, 43, 59, 60

CSI Channel State Information 19

CVAE Convolutional Variational Autoencoder 15

DCNN Deep Convolutional Neural Networks 15

DL Deep Learning 2, 6

DSCNN Depthwise Separable Convolutional Neural Network 3, 5, 22, 24, 25, 32, 38, 41, 43, 59, 60

ECG Electrocardiography 9

EMG Electromyography 9

FMCW Frequency- Modulated Continuous Wave 7

GAN Generative Adversarial Network 14, 15

GPS Global Positioning System 8, 12, 28

HAR Human Activity Recognition 1–3, 5–7, 9, 11–14, 18–23, 59, 61

HOF Histograms of Optical Flow 17

HOG Histogram of Oriented Gradients 17

IMU Inertial Measurement Unit 34

IoT Internet of Things 2

IR Infrared 15, 22

LSTM Long Short-Term Memory 11, 17, 19, 20, 61

MAC Multiply-accumulate 41

MBH Motion Boundary Histograms 17

MCU Micro-Controller Units 18

ML Machine Learning 2, 3, 6, 22, 23

MLP Multi-Layer Perceptron 20

NMSE Normalized Mean Square Error 15

PTQ Post Training Quantization 3–5, 22, 26, 28, 30, 32, 38, 41, 44, 51–53, 55, 58, 60

QAT Quantization Aware Training 3–5, 22, 25, 26, 30, 32, 38, 41, 44, 51–53, 55, 58–60

ReLU Rectified Linear Unit 25

RF Random Forest 19, 20

RGB-D RGB-Depth 8, 15

RNN Recurrent Neural Network 19, 61

ROC Receiver Operating Characteristic 55

SDK Software Development Kit 12

SIFT Scale Invariant Feature Transform 17

SoC System on Chip 61

SRAM Static Random Access Memory 26, 28, 32

STFT Short-Time Fourier Transform 14

SVM Support Vector Machines 16

TCN Temporal Convolutional Network 11

TDD Trajectory-pooled Deep-convolutional Descriptors 17, 21

TinyML Tiny Machine Learning 1–6, 18, 22, 30, 33, 38, 60, 61

UWB Ultra-Wideband 7

WISDM Wireless Sensor Data Mining 35

YOLO You Only Look Once 16

Chapter 1

Introduction

This chapter introduces the task of Human Activity Recognition (HAR) from multimodal data sources. We then reframe HAR in the context of Tiny Machine Learning (TinyML) and discuss the unique challenges, positive aspects, and motivations behind TinyML based HAR. We then define the scope of our work and highlight the unique contributions of this dissertation.

1.1 Problem Definition

HAR is the process of identifying common patterns of activity from signals collected by embedded sensing devices. HAR has a wide area of applications including smart homes where users can automate their work activities without physically being available, smart healthcare to observe patients without professional supervision, tracking the actions of the elderly, and similar applications including security and surveillance where the digitization of tasks is necessary based on an individual's activity [1, 2]. For instance, any variation from normal tasks can be easily detected by keeping an eye on everyday activities since most of the serious diseases have symptoms that force humans to deviate from their usual activities.

Due to the proliferation of portable and stationary embedded systems in human environments, it has become incredibly easy to collect and process human activity data in huge volumes. In most cases, the activity recognition systems use data from sensor modalities or vision modalities [3]. In sensor modalities, the data is collected from a single sensor such as an accelerometer/gyroscope or the combination of multiple sensors integrated with an inertial measurement unit including an accelerometer, gyroscope, magnetometer, and microphone [4]. The sensor-based HAR frameworks ensure cheap and flexible solutions while preserving privacy issues. The major issue of the sensor-based HAR platform is that many of the users are not interested to wear such bulky devices in their different body positions. In vision-based modalities, RGB video and depth cameras are exploited to perceive the information from the users [5]. In this scheme, the users are not required to wear any devices. The major drawbacks of the vision-based HAR platform include the privacy issues of the users and it is influenced by the environmental conditions. Moreover, the vision-based HAR framework requires a huge amount of memory to store high-resolution data. When these systems are deployed in multi-user scenarios such as care homes, hospitals, public spaces, etc. in a centralized server-client architecture, there is a huge pressure on compute resources and network bandwidth.

Deep Learning (DL) and Machine Learning (ML) have served as key components for various real-time applications including HAR, object and image recognition, health monitoring, surveillance and entertainment systems, and natural language processing [6, 7, 8]. In most of the use cases, traditional machine learning approaches [9] have been widely used that input the handcrafted features to a classifier network for activity recognition. Although some of the previous works [10, 11] obtained some promising results, some drawbacks regarding the handcrafted features include poor generalization, requiring prior domain expertise to extract significant features, and needing numerous trials to find the best patterns. Hence, the researchers have turned to deep learning techniques [12, 13] to recognize human activities where relevant features are automatically learned during the network training phase. Though the deep learning approaches have shown remarkable performance for HAR, the majority of the deep learning models have a huge number of trainable network parameters. In the case of the limited number of data, the excessing number of trainable parameters makes the network prone to overfitting, thus decreasing the performance of generalization [14]. Another big challenge for HAR in wearable devices is that they have limited resources and limited power availability. In such scenarios, large-sized deep learning models are difficult to deploy on edge devices [15].

1.2 Motivation

TinyML is a rapidly expanding area of machine learning approaches and applications, including hardware, software, and algorithms, able to perform on-device sensor data analytics at incredibly low power, facilitating a wide range of always-on use-cases, and aiming battery-operated systems [16]. TinyML platforms are gradually being deployed for a variety of commercial purposes and novel solutions, while substantial improvement on methods, networks, and architectures is being achieved simultaneously. TinyML and Internet of Things (IoT) in conjunction has the potential to deliver cheap, accessible and always-on solutions to various problems. TinyML uses methods such as neural architecture search to develop more sophisticated models for edge devices, and prunes, quantizes, and distils pre-trained machine learning architectures for effective on-device inference [17]. The main purpose of TinyML is to exploit cross-layer design methodologies and apply machine learning inference to ultra-low-power devices including custom-designed circuits or embedded systems that use less than a milliWatt of power [18]. When ML inference is relocated to the edge, several fundamental difficulties emerge, including latency, reliability, power consumption, limited computational resources, limited space, robustness, throughput, and privacy. Hence, it is vital to assess and explain trade-offs of popular HAR platforms mentioning the complexity of the design methodologies and resource utilization in low-power devices. Benchmarking for HAR is highly required to unlock the full potential of TinyML and guide future research in this research domain. While designing TinyML models, the challenges should be addressed and the performance must be kept at reasonable levels depending on the nature of the applications.

1.3 Scope of the Work

This research is focused on exploring and innovating novel TinyML approaches for standalone HAR systems in resource constrained edge devices. This research provides a benchmark for HAR systems in three commodity edge devices using five publicly available datasets. This research also presents a novel two-stream multi-resolution fusion architecture for HAR from video modality - increasing model performance while decreasing model size.

1.4 Contributions

The major contributions of this thesis are two folds. Firstly, we are the first research study to specifically experiment and present the benchmark for HAR using standard Convolutional Neural Network (CNN)s and Depthwise Separable Convolutional Neural Network (DSCNN)s in resource constrained edge devices. A summary of the technical contributions of this part is as follows:

- We systematically tested multiple CNN and DSCNN structures on five popular HAR datasets with two different data modalities: vision and time-series data.
- We tested the performance of the models in terms of accuracy, precision, recall, F1-Score number of parameters, and model size.
- We tested the effectiveness of two different quantization approaches: Quantization Aware Training (QAT) and Post Training Quantization (PTQ) in terms of their effect on model size, number of parameters, and model performance.

- We deployed the tested models into standalone applications in three publicly available resource-constrained edge devices.
- We tested the performance of the deployed standalone applications in terms of inference latency and power consumption.

Secondly, we proposed a novel two-stream multi-resolution fusion architecture for human activity recognition from video data in resource-constrained edge devices. The two streams of the model are called the context stream and the fovea stream. The context stream takes a resized image of the full-size input and the fovea stream takes the cropped center as input from the resized image. Due to camera bias, subjects in activity videos are often in the center of the frames. By focusing on the cropped center image with the fovea stream, important spatial features can be extracted. Additionally, using the two-stream architecture reduces the dimensionality of input data. The extracted features from the two streams are fused using the concatenation operator and sent to a fully connected layer for recognition purposes. The performance of the proposed fusion architecture is evaluated in two challenging video datasets: KTH and UCF11. We also performed ablation studies to validate our proposed model structure. Two quantization methods: PTQ and QAT were performed on the proposed model to prepare them for deployment. We deployed the quantized models into three commodity edge devices and reported their performance in terms of inference time and power consumption. To the best of our knowledge, this is the first research work that develops a two-stream multi-resolution fusion architecture for human activity recognition in edge devices utilizing TinyML based on the existing state-of-the-art literature in this research domain. Moreover, this work employed video data for human activity recognition from a TinyML perspective for the first time. The summary of our main contribution to this part is as follows:

- We proposed a novel two-stream multi-resolution fusion architecture for efficient human activity recognition from video data.
- We proposed two streams for the fusion architecture: the context stream and the fovea stream. The input images are resized to a lower resolution and input into the context stream. The fovea stream takes the cropped center portion of the lower-resolution resized image as input.
- We quantized the proposed model using two quantization methods: PTQ and QAT for deployment.
- We deployed the quantized versions of the proposed model into standalone applications in three commodity resource-constrained edge devices.

- We tested the proposed model performance in two challenging human activity video datasets in terms of the number of parameters, accuracy, precision, recall, F1-Score, and size. Moreover, we evaluated the performance of the deployed quantized models in terms of their inference time and power consumption.

1.5 Thesis Organization

The rest of the thesis is organized as follows. Chapter 2 describes the most recent literature on HAR from multimodal data sources: multimodal sensing devices, smartphone sensor data, radar signal, and image and video data. The chapter also includes the most recent literature on TinyML benchmarking efforts and lightweight deep learning architectures for HAR on wearable devices. Chapter 3 presents the benchmarking procedure, the proposed multi-resolution fusion approach, the description of the datasets, the CNN architecture, the DSCNN architecture, the PTQ and QAT methods for quantization, the description of the tiny edge devices, the standalone application architecture, and the demonstration of the TinyML development and deployment lifecycles. The experimental findings and discussions are demonstrated in Chapter 4 which includes the benchmarking results, quantitative results analysis of the proposed multi-resolution fusion architecture, ablation studies, and the performances of both the benchmark models and proposed architecture in commodity tiny edge devices. Lastly, the conclusion, limitations, and potential future research works are highlighted in Chapter 5.

Chapter 2

Literature Review

In this chapter, firstly, we explore the traditional approaches behind HAR from multimodal data sources. The traditional related works are divided into four sub-sections based on the types of sensing devices: multimodal sensing devices, smartphones, radar devices, and image and video devices. We discuss the strengths and weaknesses of all reviewed works. We then explore the current literature on TinyML benchmarks and lightweight deep learning models of HAR from various data sources.

2.1 Overview on Human Activity Recognition

HAR plays a significant role in the everyday life of people because of its ability to learn extensive high-level information about human activity from wearable or stationary devices. A substantial amount of research has been conducted on HAR and numerous approaches based on deep learning have been exploited by the research community to classify human activities. Recently, DL algorithms have become more popular due to their automatic feature extraction capability from vision or image data [19] as well as time-series data [8] that enables the learning of high-level and meaningful features. DL techniques have been generally outperforming traditional ML approaches for activity recognition in terms of classification performance measures such as accuracy, precision, recall, and F1-Score [20, 21]. The recognition of human behavior based on deep learning architecture, especially CNN is a composite system that is comprised of several key stages. An overall system architecture of a typical CNN-based activity recognition system is illustrated in Fig. 2.1. The first stage is comprised of the selection and implementation of sensing devices. Data collection is the next step where an edge device is used to perceive data from input

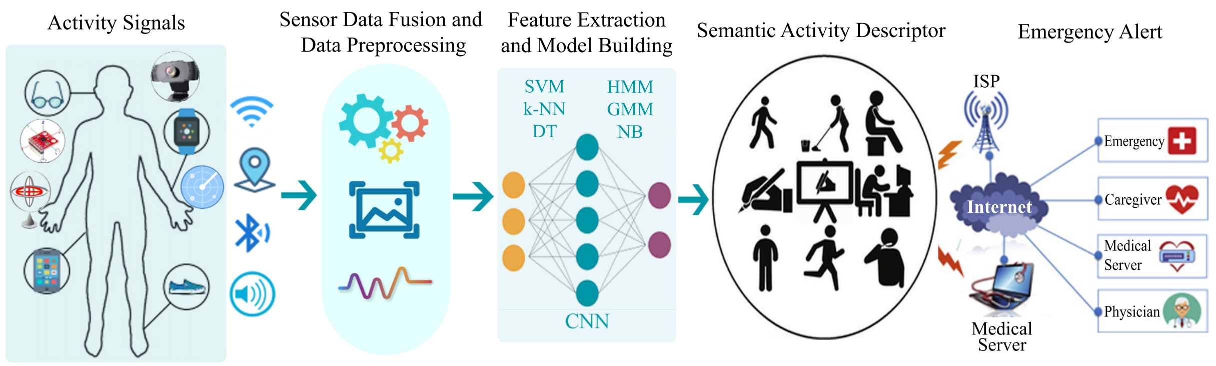


Figure 2.1: An overall system architecture of CNN-based human activity recognition.

devices and transfer it to the main server through various communication systems such as Wi-Fi and Bluetooth. The deployment of computing and storage resources at the point where data is being collected and processed is referred to as edge computing which incorporates sensors for data perception as well as edge servers for reliable real-time information processing [22, 23]. The feature extraction and selection stage extract the necessary features from the raw signals; this stage is performed automatically in the case of CNN; no hand-crafted feature extractions are required. This stage contains the CNN architecture or variants of CNN structure for the recognition of activities. The last step includes a notification system through which an agent (human or machine) can be notified. The notification system can aid in emergency scenarios by notifying emergency contact personnel or emergency services.

With the rapid growth and performance improvement of deep learning techniques particularly CNN architectures, numerous researchers have adopted them for dealing with HAR problems. This section is focused on reviewing the four commonly used categories of devices for HAR: i) Human activity recognition from multimodal sensing devices, ii) Human activity recognition from smartphone sensor data, and iii) Human activity recognition from radar signals, and iv) Human activity recognition from image and video signals - as well as the CNN tools that have been used in conjunction with these devices for performing activity recognition. Figure 2.2 illustrates the human activity recognition systems and their different types based on the sensing devices, signals, or sensing modalities. The multimodal sensing signals based human activity recognition systems generally use data from the following sensing modalities: accelerometer, gyroscope, magnetometer, and barometer. Smartphone-based human activity recognition systems utilize smartphone sensors to collect and classify data. Radar-based human activity recognition systems use various types of radars to collect data. Popular radar variants include Doppler radar, Frequency-Modulated Continuous Wave (FMCW) radar, interferometry radar, and Ultra-Wideband (UWB).

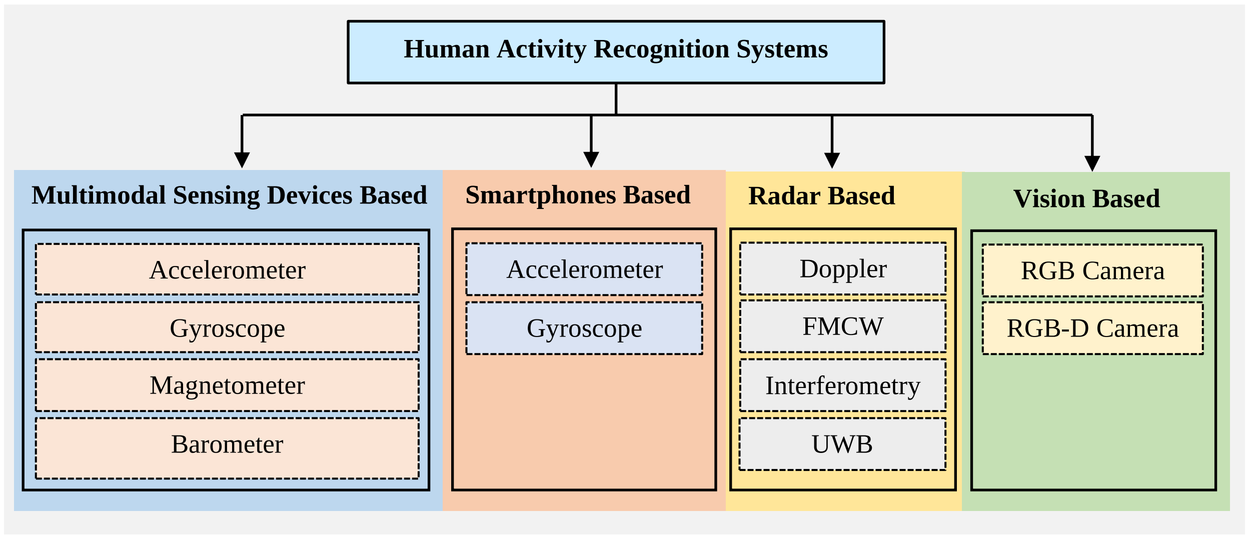


Figure 2.2: Human activity recognition systems and their modalities.

Vision-based systems collect data through RGB cameras or RGB-Depth (RGB-D) cameras. The recent works for human activity recognition are described below where we discuss the data collection strategies of each modality and the developed systems for each modality subsequently. In the next few sub-sections, we discuss the four categories of devices and how the CNN-based models and tools are utilized to infer human activity from captured data using those devices.

2.1.1 HAR from Multimodal Sensing Devices

In recent years, the research for activity recognition is focused on the combination of multiple sensor data (accelerometer, and gyroscope) that may increase the reliability of the developed system in certain cases [24, 25].

We consider various embedded body-worn solutions such as smart watches, smart necklaces, bands, helmets, and watches containing sensors like 3-D accelerometer, gyroscope, magnetometer, and barometer excluding smartphones as multimodal sensing devices [26]. These solutions are generally smaller than modern smartphones and require less power to run. However, these solutions oftentimes do not have Global Positioning System (GPS) and network connectivity. These devices generally do not require everyday charging. As these devices are always worn or kept on a person by the users, the sensors can record human motion and process them. Human activity generates acceleration and angular velocity. 3-D accelerometer sensor can sense acceleration along the 3-axes. A gyroscope and magnetometer can be used to sense the angular velocity and orientation. A barometer helps in sensing height changes during activities. Combining these properties, multimodal sensing solutions can infer human activities.

Wearable sensors have become increasingly popular in a wide range of applications as they can provide accurate and reliable data on daily human activities to ensure an ambient assisted living environment for the elderly [27]. Wearable sensors are capable of perceiving human body movements directly and efficiently over a long duration. The rapid growth of wearable technology has enabled the development of several types of smart sensors to capture physiological parameters accurately with lower power consumption and fewer processing resources. These sensors can be easily integrated into smartphones, bands, watches, and even clothes. A brief description of a few sensors that are commonly used for HAR is given below.

The most popular and commonly used sensor for HAR is an accelerometer [28]. An accelerometer is a sensor utilized to determine the acceleration - which is the rate of change of the velocity of an object. The sampling frequency of an accelerometer typically falls between tens and hundreds of Hz. There are numerous types of accelerometers on the market that use variable capacitance, piezoresistive, or piezoelectric transduction techniques. The operating concept of almost all types of accelerometers is the same: a mass reacts to acceleration by forcing a spring or an equivalent component to expand or compress according to the measured acceleration. A gyroscope [29] is also a frequently used sensor along with the accelerometer and mounted on the same body parts for human activity recognition. A gyroscope is a sensor for measuring angular velocity and orientation. Similar to an accelerometer, the sampling rate of a gyroscope sensor ranges from tens to hundreds of Hz. As a gyroscope has three axes, it also offers three separate time sequences. Another commonly used wearable sensor for HAR is a magnetometer [30] that is integrated with an accelerometer and a gyroscope into an inertial measurement unit. This sensor determines the change of a magnetic field at a specific position. The sampling frequency of this sensor is similar (tens to hundreds of Hz) to the accelerometer and gyroscope. A magnetometer sensor has three axes as well. Electrocardiography (ECG) [31] is a biometric sensor for HAR that detects the electrical signals produced by the heart. The information about the rate and regularity of heartbeats is extremely useful provided by the ECG signal. It is challenging to analyze subject variations in the ECG data because everyone's hearts vibrate in their unique ways. The output of an ECG sensor is univariate time-series data. Electromyography (EMG) [32] sensor is used for human activity monitoring that detects muscle response or electrical signal in response to a nerve's stimulation of the muscle. Both the EMG and ECG sensors are required to be attached to human skin to record the data. Even though EMG is less frequently employed in traditional situations, it is more suited for fine-grained motions including hand or arm movements, and facial expressions. The EMG sensor generates the output in the shape of univariate time-series.

In multimodal sensing devices, a big challenge is learning the inter-modality correlations along with the intra-modality data for CNN-based human activity recognition. To solve this issue, some CNN-based approaches have been developed that combine various modalities for the development of single extracted features or ensemble the output of different architectures. The simplest way to handle multimodal sensor data is combining the data from all sensors ignoring the sensor modalities although it has a chance of losing accurate correlations. In order to capture local and spatial dependence over time and sensors, respectively, a multi-modal CNN architecture is proposed in [33] that used 2-D kernels in both convolution and pooling layers. As the relationship between the non-adjacent modalities is absent from traditional CNN, the system developed in [34], proposes a novel architecture in which any sequence of signals can be adjacent to every other sequence. Several systems have already been developed that consider each sensing modality initially and then combine them. This architecture provides modality-specific information along with versatile distribution of complexity. An architecture named EmbraceNet was proposed in [35] that processes the sensors' data separately and feeds them into the EmbraceNet. An effective way of fusing multimodal information using this architecture is through docking and embracing structure. A deep multimodal fusion architecture is introduced in [36] that calculates the confidence score of each sensor automatically and combines the features of multi-modal sensors based on the scores.

Some of the frameworks are introduced to minimize the interference between the used sensors by treating each sensor axis independently. In this case, 1-D CNNs are very popular for feature learning of each separate channel. A 1-D CNN architecture, proposed in [37] that omitted the pooling layers to achieve more detailed features where the data from the accelerometer and gyroscope are fed into the network separately. In [38], the acceleration of the accelerometer and gyroscope from seven different body positions are used to produce frequency images that are served as the input of two-stream CNN to learn inter-modality features. Very recently, a few CNN-based systems have been developed to handle univariate multichannel time-series data where the same sized filter is implemented to all time sequences. In this scenario, the raw signal is converted to a 2-D array by stacking along the modality axis which is then applied to 2-D CNN with 1-D filters [39]. However, the characteristics of all sensor data do not combine externally, but they interact through mutual 1-D filters.

There is another new trend in the area of deep learning called attention mechanisms [40] that has become a very popular and frequently used concept in diverse application domains including human activity recognition in recent years. In the current scenarios, the majority of the developed systems used shallow feature learning architectures that could not recognize human activities accurately in real-world situations. To solve this issue, Hamad et al. [41] used a dilated causal convolution with multi-head self-attention for physical activity recognition. During recognition, the multi-headed self-attention is utilized to allow the model to highlight significant and vital time steps rather than irrelevant time steps from the sequential feature space. The proposed architecture obtained an F1-Score of 92.24% from the experimental findings. Wang et al. [42] introduced an activity recognition system that processed weakly labeled information utilizing attention mechanisms. The compatibility between global features and local features is computed using this approach. By weighing their compatibility, the attention mechanism enhanced the salient activity data and suppressed the insignificant and slightly confusing data. The experimental results revealed that the scheme appraised an accuracy of 93.83%. Tan et al. [43] presented faster region-based CNN and attention-based Long Short-Term Memory (LSTM) for human activity recognition where the CNN extracted the feature as posture vector and the Bi-Directional Long Short-Term Memory (Bi-LSTM) architecture classified the human activities. An attention layer is added between the two Bi-LSTMs. The developed network obtained precision of 97.02% and recall of 96.83% from the experimental findings.

In the current state-of-the-art, most of the existing frameworks for HAR take the global information of input sequence and avoid local information that demonstrates changes in behaviors, causing the method to be responsive to external factors including occlusion and illumination change. To resolve this problem, some of the studies [44, 45] consider the local spatial features, global spatial features, and temporal features for HAR. Andrade-Ambriz et al. [46] proposed a human activity recognition framework using a Temporal Convolutional Network (TCN) that utilized spatio-temporal features as input of the architecture. The experiment demonstrates that the developed prototype achieved 100% precision and recall for two popular datasets. The scheme shared the activity recognition results to a robot called NAO during real-world environment testing. Zhu et al. [47] introduced a multimodal activity recognition scheme that fused three spatial features: local, global, and temporal features of input signals to classify different human actions. The proposed system divided the input into three segments where the global spatial features are found from the first segment (RGB frame) using a spatial CNN, the local spatial features are extracted from the local blocks utilizing another spatial CNN, and the last segment (optical flow) is utilized to extract temporal features through the use of a temporal CNN. The three architectures are evaluated individually using two benchmark datasets and the final output is obtained using the weighted sum of the three networks. The best accuracy of 94.94% is found from the experimental results for the UCF101 dataset. Gao et al. [48] proposed a framework called DanHAR that combined channel and temporal attention on residual networks to enhance feature representation capability for human activity recognition. The proposed architecture takes a time window as input and sends it to convolutional layers to obtain visual features. This network then generates channel attention through pooling layers (max-pooling and average-pooling) to combine features along the temporal axis. It is found from the experimental results that the proposed architecture obtained an accuracy of 98.85% for the WISDM dataset. In another research, Tang et al. [49] developed a triplet cross-dimension attention model for HAR, which introduced three attention parts to make the cross-interaction between sensor, temporal, and channel dimensions. The performance measure shows that the F1-Score of 98.61% is obtained by the developed system. It is worth mentioning that the system is tested in a real-time environment using a Raspberry Pi prototype.

2.1.2 HAR from Smartphone Sensor Data

Smartphone has become very popular for HAR thanks to the rapid growth of modern technology as it has various built-in sensors for this task [50]. The major problem of the traditional wearable sensing device is that the users should carry an extra device; sometimes they are not willing to carry it, or a few times get forgotten. As almost everyone now has a smartphone, it has become an excellent choice to conduct research using smartphone sensor data, which ensures the portability of the developed systems to a great extent [51].

All modern smartphones contain sensors such as accelerometer, gyroscope, and magnetometer. Smartphones generally require more power than multimodal sensing solutions. However, almost all smartphones require regular daily recharging. Smartphones have more processing powers than multimodal embedded solutions. Smartphones also have GPS and network connectivity, making them viable for transferring data and decisions in various client-server models [52]. Thus, smartphones have access to more sophisticated data-heavy models. Users generally carry smartphones on locations such as thigh, chest, and hand. As human motion generates acceleration and angular velocity, smartphone applications can sense these changes through the embedded smartphone sensors and process them. Smartphones also provide high-level access to sensor data via the Software Development Kit (SDK) of the operating systems. SDK also provides additional support such as always-on applications, notification and alarm systems, and sensor data buffers.

One of the major issues that should be considered is the position of the smartphone as it can be kept in a pant pocket, hands, bags, and shirt pocket. It is evident that due to the various locations of the smartphone, the raw signals change considerably as the movements of the various parts of the body are different. To handle this problem, some of the reviewed literature developed position-independent solutions. A smartphone-based position-independent activity recognition system using CNN was introduced in [53] that used time-domain statistical features. Here, mean-centering is used to convert the raw input to an appropriate form to train the optimum threshold without any bias. A 1-D CNN is introduced in [54] that used smartphone accelerometer data from different body positions like the bag, hand, and pocket for activity recognition. As the data for different body positions are used, the developed system effectively ensures the position-independent property. Another big challenge for smartphone-based systems is that traditional deep learning architectures cannot be easily embedded in such systems due to low power capacity and limited computational capacity. To resolve this issue, a few of the systems have been developed that merged the hand-crafted features and deep features for activity recognition. Decreasing filter size is a potential solution to reduce the size of the network that optimizes computing operations. A HAR system is introduced in [55] where the deep features and hand-crafted features are arranged in parallel, and the features are then incorporated into the 1-D CNN architecture. The performance of the developed system has been increased with a small number of computational operations. The system developed in [56] used just one CNN layer and two fully connected layers where the spectrogram features are fed into the network for activity recognition. The experimental findings revealed that the system achieved milliseconds to tens of milliseconds of computing time for a single prediction.

In another study, the authors of [57] proposed an attention-based multi-head architecture for HAR. The developed architecture had three lightweight convolutional heads; each is designed to extract features from collected data using 1-D CNN. The lightweight multi-modal architecture is stimulated with an attention mechanism to improve CNN's representation capability, enabling the automatic selection of significant features while suppressing irrelevant ones. Although two datasets have been utilized here, the highest accuracy, precision, recall, and F1-Score of 98.18%, 97.12%, 97.29%, and 97.20%, respectively are achieved from WISDM data. Zhang et al. [58] developed a system that combined the concept of CNN and attention mechanism for activity recognition using the data from a smartphone. Here, the attention is incorporated into multi-head CNNs that facilitate extracting and selecting features efficiently. The proposed scheme achieved accuracy and F1-Score of 96.4% and 95.4% from the experiments. The author of [59] introduced a novel deep learning architecture called LGSTNet, combining the concept of CNN and attention mechanism to recognize human activity from the data of accelerometers and gyroscopes. The activity window is fragmented into various sub-windows in this system, and the local spatial-temporal attributes from those sub-windows are learned using an attention mechanism and CNN. It is evident from the experiments that the obtained F1-Score of the proposed network is 95.69%.

In another research, Nafea et al. [60] presented a HAR framework that used spatial information and temporal information, extracted by CNN with differing kernel sizes and Bi-LSTM, respectively. The retrieved spatio-temporal data were merged in a mixed model that was trained and verified using two benchmark datasets, yielding a 98.53% accuracy, and Cohens Kappa of 98% for the WISDM dataset. Nair et al. [61] developed a temporal convolution network-based architecture to recognize human activities from raw motion data collected utilizing smartphone sensors. To deal with sequence information with big receptive fields and temporality, dilations and causal convolutions have been developed in this system. The accuracy and F1-Scores of 97.8% and 97.7%, respectively are achieved from the experimental results using the encoder-decoder temporal convolutional network.

2.1.3 HAR from Radar Signal

The current research is focused on the device-free approach as it does not include any devices to take while participating in any activities. To ensure a device-free solution, a radar-based system shows the best performance due to its insensitivity to daylight and environmental effects as well as contactless-manner [62]. Radar signal-based sensing modality is widely used for stationary surveillance. Radar or radio detection and ranging systems detect both living and inanimate objects through reflection [63]. Radar systems generate intermittent high-frequency radio waves and transmit them to the environment around them. Radio waves are electromagnetic waves that travel at the speed of light. After hitting objects in the environment, radio waves bounce off or reflect from them. Radio systems can receive the reflected radio signal and extract properties of the object including size, shape, distance, and movement from the time required to detect the reflection and the change in frequency due to collision. Radio signal-based sensing modalities deployed in surveillance situations can thus detect stationary objects, humans as well as human motions through the characteristics of the received signal and infer the human activity.

In general, the radar signal is converted to the time-frequency domain which is a separate part from the learning architecture that sometimes does not extract the optimal features. In some cases, the raw signal is transferred to the Short-Time Fourier Transform (STFT) or 2-D matrix and the deep learning architecture treats the 2-D matrix as an optical image. However, the optical image pixels have high spatial correlations, whereas the 2-D radar matrix pixels have a lot of temporal correlations. Hence, treating them as the same is not optimal for classification purposes. To improve the performance of the radar-based systems, some researchers focused on the use of variants of CNN rather than conventional CNN to resolve these issues. An end-to-end network named RadarNet is introduced in [64] where the STFT is substituted by two 1-D convolutional layers. The developed system merged all the steps (micro-Doppler radar data representation, extraction of features, and classification) of HAR in a single architecture. F-ConvNet, another end-to-end architecture is proposed in [65] that used three convolutional layers for multi-scale feature extraction. A novel layer named Fourier layer is proposed here that includes Fourier initializations and two branches of processing for learning the real and imaginary segments individually. In addition, to improve the classification accuracy, dilated convolutions are used. To reduce the computation complexity, an end-to-end network (1-D CNN) is designed in [66] for activity recognition using radar signals. The proposed system used the inception densely block (ID-Block) that is customized for the proposed 1-D CNN where ID-Block is comprised of an inception module, network-in-network methods, and a dense network.

To solve the issue of limited training data, Generative Adversarial Network (GAN) are frequently utilized in recent times. In [67], a GAN is developed for HAR using micro-Doppler signatures of radar. While the GAN is trained with the original micro-Doppler images, it generates a lot of similar images like the original that are fed into traditional CNN for training. The use of the increased number of images enhances the performance of the developed system. Erol et al. [68] proposed a human activity recognition platform that used synthetic radar data generated by GANs. The system introduced an Auxiliary Classifier Generative Adversarial Network (ACGAN) to generate a large number of training samples. The average test accuracy of 82.56% is found from the experimental findings using the proposed GAN networks with Deep Convolutional Neural Networks (DCNN) architecture. The main purpose of the developed framework is to reduce the classification confusion among similar activities by increasing the size of the training dataset. The authors of [69] introduced a framework to use the discriminator of the GAN for human activity recognition using limited radar data. To initialize the parameter of the GAN, a Convolutional Variational Autoencoder (CVAE) is utilized in the proposed system. From the experimental study, it is found that the scheme obtained an accuracy of 94.89% from a few numbers of data samples. In another work, Alnujaim et. al. [70] used GAN to increase the time-frequency images retrieved from a single angle into images from numerous angles resulting in the improvement of the sample set from multiple viewpoints. The average Normalized Mean Square Error (NMSE) of the developed prototype is 0.50E-04. However, the developed framework generated data samples with high similarity due to the convergence instability of GAN.

2.1.4 HAR from Image and Video data

Due to advances in technology, both RGB and RGB-D cameras are easily accessible and cost-efficient. Vision-based systems are effective for the surveillance of large regions in an effective manner. Image and video sensing modalities are more accessible and easier to setup than radar sensing modalities [71]. Even cheap RGB cameras nowadays have night vision capability through Infrared (IR) sensing. As these systems are stationary solutions, very simple techniques such as background subtraction can be used to localize and monitor motion in the surveilled area. The detected motions can then be passed to CNN models to infer human activity. With the advancement of specialized processors for neural network processing, RGB solutions with built-in neural network processors and network connectivity are available. These systems can capture images or video sequences and process them locally using saved deep learning models in offline mode or can send data to servers running more sophisticated models in online mode.

Although previous research advances were focused on traditional vision-based algorithms, current advances in both deep learning algorithms and hardware enabled us to deploy highly effective deep learning algorithms alongside vision systems. Computer vision-based human activity recognition systems face several challenges such as interclass variation and intraclass similarity, diverse and complex backgrounds, multi-subject interactions, group interactions, videos from long distances, and low-quality images. Intraclass variation arises from separate people performing the same action in their own ways. Interclass variation arises from the numerous types of activity we perform in our day to day lives. Vision-based image and video datasets also have various types of backgrounds. Backgrounds differ in lab scenarios as well as in real-world scenarios. Image and video data also suffer from inherent complexities such as pixilation, aliasing, light level differences, viewpoint variations, and occlusions [72]. Video-based human activity recognition datasets are more common than still image-based datasets, as activities are regarded more as a sequence of actions than a one-off scenario.

Most of the available human activity recognition datasets contain video sequences of Activities of Daily Life (ADL). The system in [73] introduced an abnormal human activity dataset containing abnormal actions such as coughing, chest pain, faint, vomiting, and taking medication while also implementing a real-time high-speed recognition algorithm based on the You Only Look Once (YOLO) architecture [74]. However, this system only works in cases when a single human is monitored. The system cannot recognize group activities, overlapping objects, and small objects due to the spatial constraints of the YOLO backbone.

While performing human action recognition from video sequences, initial CNN-based research works processed all of the frames of a video for recognizing tasks represented in the video. However, this was inefficient due to the huge time, computation, and memory required to process all the frames. An alternative solution is proposed in [75] where only a selected number of frames of a video are classified instead of all the frames of a video. This drastically reduced the computation time and computation requirements, while also making the system real-time. In general, 30 frames from a video are selected for classification in a deterministic way based on the total number of frames in the video. These selected frames are classified to determine the represented actions and their confidence levels. These confidence levels and actions are averaged to determine the final action and confidence level of the entire video. However, this system works best when a video sequence contains a single action group. When a video contains multiple actions or action groups, determining a single action across an entire video becomes problematic. The CNN methods discussed till now only take the spatial domain characteristics of the videos into consideration. However, temporal domain characteristics can also be extracted from the videos which might act as discriminating features. A two-stream CNN for human action recognition is introduced in [76]. The spatial domain stream is trained on the individual frames of the videos. The temporal domain stream is trained on stacked multiple-frame dense optical flow. The dense optical flow in the temporal domain contains motion data of the objects. The two separate streams are combined by calculating stacked L2 normalized SoftMax scores as features. These features are classified using multi-class linear Support Vector Machines (SVM). The main issue with this method is the huge memory requirements to store all the optical flow data. To reduce the size of the saved data, float point data were converted to integer point data, and the saved data was compressed using JPEG. Despite these measures, the saved optical flow data took huge memory spaces for even smaller datasets. For huge datasets such as the YouTube 1M [77], YouTube 8M [78] this method would require big data storage solutions. An alternate solution for taking the features from the temporal domain into consideration is presented in [130]. As opposed to training two separate 2-D CNN networks in their respective spatial and temporal streams, the system developed in [79] used 3-D convolutions to extract features from both the spatial and the temporal domain. In this way, a single 3-D CNN can be used in place of two-stream two CNN solutions. This also effectively solves the data storage issue of [76]. The performance of this model is comparable to the two-stream two CNN solutions. However, this model requires more labeled data than unsupervised counterparts, and thus, for large video human action datasets, accurate labeling poses a big challenge.

Handcrafted features such as the Histogram of Oriented Gradients (HOG), Histograms of Optical Flow (HOF), Motion Boundary Histograms (MBH) have been historically used for human action recognition [80]. The framework developed in [81] introduces Trajectory-pooled Deep-convolutional Descriptors (TDD), a combination of handcrafted features and deep-extracted features for human activity recognition. The handcrafted features and the features extracted using deep 2-D CNN networks are aggregated based on trajectory constrained pooling. Spatio-temporal normalization and channel normalization are further used to transform the feature maps. The TDD features are highly discriminative and are learned automatically. However, this method performs slightly worse than two-stream 2-D CNN solutions. While the performance of TDD is excellent in the spatial domain, its performance is worse than or comparable to the performance of two-stream solutions in the temporal domain. The authors of [82] proposed a hierarchical approach for complex human recognition that used background subtraction and HOG for image preprocessing, deep learning architectures (CNN, and LSTM) for feature selection and to maintain the previous data, and Softmax-k-nearest neighbors for classifying the human activities. The proposed architecture has been evaluated using the UCF101 dataset that contains 101 complicated human activities. Extensive experiments revealed that the developed system achieved an accuracy of 93.80%. In another research, a fusion of Scale Invariant Feature Transform (SIFT) and optical flow method are exploited [83] to extract the shape, gradient, and orientation features from videos of human action datasets. Additionally, CNN is used to recognize human activities by training and testing the datasets. Two popular datasets, namely Weizmann dataset and KTH dataset are utilized to evaluate the performance of the developed framework and appraised an accuracy of 98.03%, and 94.96%, respectively.

2.2 Literature on TinyML Benchmarks

Several benchmarks have been conducted regarding TinyML in various research directions. MLPerf [84] is a benchmarking suite for machine learning inference with aims to add power measurements. However, the current MLPerf inference benchmark excluded Micro-Controller Units (MCU) and other low-power devices due to the lack of tiny benchmarks and appropriate deployments. In another research, the deep learning benchmark called MLPerf Tiny [85] is designed for integrated and ultra-low-power systems. This benchmark focused on four major application areas including keyboard spotting, visual wake words, image classification, and anomaly detection. Further, Sudharsan et al. [86] used three fully connected neural networks and each of the networks is trained using 10 popular datasets. This benchmark demonstrated the onboard performance of the models on 7 widely used MCUboards that are the major components of TinyML hardware. In another study, Profentzas et al. [87] utilized three deep learning libraries: uTensor, TF-Lite-Micro, and CMSIS-NN, to develop a benchmark for analyzing the trade-offs of low-power IoT devices. Two popular datasets namely MNIST and CIFAR-10 are used for the benchmark study. From the experimental findings, it is revealed that the CMSIS-NN is the most efficient among the other frameworks in terms of energy consumption for the specific test scenarios. Furthermore, Pau et al. [88] highlighted two application areas such as human activity recognition and anomaly detection for benchmarking. The system developed hybrid binary neural network architecture that is the combination of 32-bit integer layers and binary layer. The input and output layers in this network have the same weights, i.e. 32-bit, however the intermediate layers are designed utilizing binary weights. The system demonstrated how the quantization layer selection is crucial for reducing network size and complexity without affecting the accuracy of the architecture.

2.3 Literature on Lightweight Deep Learning Models for HAR

Nowadays, deep learning has become a prominent research topic in various real-time applications including human activity recognition due to the rapid increase of mobile computing devices. To ensure real-time applications, many researchers are focusing on developing lightweight deep learning architectures that are applicable to tiny embedded devices [89, 90]. Hence, deep learning on tiny edge devices has become a research concern over the past few years. The lightweight deep learning architectures that have been developed for human activity recognition are briefly demonstrated below.

Gupta et al. [91] developed a TinyML system for HAR by optimizing the existing deep learning architectures. The system used three variants of deep learning architectures such as simple

CNN, DeepConv LSTM, and multi-layer LSTM. To reduce the weight of the models, the proposed system exploited two popular optimization techniques including pruning and quantization. For the evaluation of the developed platform, the system is tested with a benchmark dataset called UCI-HAR. The obtained accuracy of the optimized model (DeepConv LSTM) is 90.48% which is relatively low than the original model, but the size of the optimized architecture decreased drastically. However, the developed scheme has not been tested with any edge computing devices. Zhou et al. [92] proposed a lightweight deep learning architecture called TinyHAR to classify activities in wearable devices. The developed framework extracted the significant features from the raw data using different multimodal saliencies, multimodal collaboration, and time–series information. DeepConv LSTM is used as a baseline architecture. The proposed system has been tested with six popular open-access datasets and the best F1-Score of 98.99% is achieved from the SKODA dataset. Compared to the original model, the TinyHAR achieved the same performance with around 6% less model sizes. However, the deployment of the proposed architecture on edge devices including Seed Studio's RISC-V based Sipeed MAix BiT platform is still ongoing. To develop a HAR with lower computing requirements and latency, Agarwal and Alam [93] introduced a lightweight Recurrent Neural Network (RNN)-LSTM architecture that is deployed on Raspberry Pi devices. The experiment is performed on the widely used Wireless Sensor Data Mining (WISDM) dataset to evaluate the performances of the developed architecture. From the experimental findings, it is revealed that the developed framework achieved an accuracy of 95.78%. However, the proposed system has been tested with only one dataset which did not prove the good generalization ability of the model.

In another research, the authors of [94] presented a lightweight human activity recognition approach called LiteHAR that does not need substantial parameters for training. In this system, the high-level patterns are retrieved from raw Channel State Information (CSI) data using randomly initialized convolution kernels without training the kernels. The activity recognition is done through Ridge regression classifier, which has linear computational complexity and is relatively fast. The evaluation of the LiteHAR architecture is done on a benchmark dataset called StanWiFi and the obtained accuracy is 91% for classifying seven ADL. Coelho et al. [95] introduced a lightweight two-level classifier utilizing pruned and quantized CNN architecture for HAR on portable devices. The extracted features from the collected data are fed into SVM to classify the static and dynamic activities. Moreover, the decision tree is exploited to recognize the activities from the extracted features and static activities. Additionally, the lightweight CNN architecture recognized the activities from the preprocessed data and dynamic activities features. The found accuracy and F1-Score are above 90% utilizing the benchmark dataset UCI-HAR. However, the performance of the developed network has not been tested with any tiny devices. In [96], the authors presented a machine learning framework based on the concept of Random Forest (RF) and CNN for HAR on microcontrollers. The developed networks are converted to C code to deploy in MCU. To evaluate the performance of the models, a popular benchmark dataset called PAMAP2 is used. The mid-range microcontroller namely nRF5340 is exploited as an edge device for experimentation. The experimental results depicted that the RF architecture obtained the similar level of accuracy for HAR but several times faster than the baseline CNN architecture.

In another work, Liu et al. [97] proposed a lightweight deep learning architecture using linear grouped convolution for HAR on portable devices. In this system, the data collected from sensors are fed into the architecture as an overall data stream. Six popular human activity recognition datasets are utilized to test the proposed network. The developed system obtained good recognition performance while decreasing the number of parameters and computation complexity. Moreover, the real-world experiments for HAR are conducted on two portable devices such as Android and Raspberry Pi. However, there are still more than half million parameters in the optimized architectures. Huang et al. [98] utilized channel equalization and channel selectivity in CNNs for HAR to reinstate the channels that imploded as a result of normalization. The proposed channel equalization technique forced all channels at the same layer to participate in feature extraction by preventing the network from depending on just a few selectively activated channels and thereby providing good generalization capability. The evaluation of the proposed architecture is done through the use of six open-access benchmark HAR datasets. The best accuracy of 99.17% is found from the USC-HAD dataset. Finally, the developed network is tested with a Raspberry Pi to get the output from real-world deployment scenarios. Lattanzi et al. [99] presented a comparative study between two types of neural networks such as Multi-Layer Perceptron (MLP) and CNN to explore some prominent performance measures needed for tiny wearable devices including inference time, power consumption, and the use of memory for HAR. In this study, the smartwatch called Hexiwear is used as a wearable device to capture the data from the participants, and the low-power Arm Cortex-M4 is exploited as a processing device. The experimental findings revealed that the MLP architecture obtained a 4 times reduction in the use of memory, and about 36 times reduction in energy consumption than the CNN model while achieving the same accuracy measure. Ankita et al. [100] developed a human activity recognition framework combining the concept of convolutional layers, LSTM, and the deep neural network. The proposed architecture retrieved the significant patterns from the raw data automatically and classified them with some model attributes. The developed framework is tested on a publicly available benchmark dataset called UCI-HAR collected through Samsung Galaxy S2 from the volunteers. It is found from the experimental analysis that the developed lightweight model obtained an accuracy of 97.89% for activity recognition. However, this scheme failed to encode the positioned orientation of the activities.

2.4 Summary

In this chapter, we explored the use of CNNs in HAR systems through the presented sensing modalities. We also presented the different and effective use of various CNN architectures and techniques such as 1-D CNNs, inception blocks, dense blocks, two-stream convolutional networks, 3-D CNNs, and specialized features such as TDD. We discussed the reviewed systems in terms of their strengths, weaknesses, and future scopes. We presented the current literature on benchmarking efforts to test these systems in edge environments. We also presented the current advances in lightweight deep learning models for HAR in edge devices.

Chapter 3

Proposed Methodology

In this chapter, we discuss the underlying structures making HAR possible in tiny edge devices. CNN's have revolutionalized modern pattern recognition tasks such as HAR. However, CNN's have huge compute and memory requirements - making vanilla CNN's unsuitable for TinyML. We discuss alternate structures such as DSCNN - that reduce the number of operations and memory requirements compared to CNN. We then discuss quantization methods: QAT and PTQ to compress both CNN and DSCNN architectures for deployment in resource constrained devices. We discuss the standalone embedded application architecture and the features of the commodity edge devices.

3.1 Benchmarking HAR

A generalization of a HAR application in the context of TinyML and resource constrained devices is illustrated in Fig. 3.1. A generic HAR application includes an initialization stage and a main loop. In the initialization stage, device pins are reset, connections to sensing devices and other services are checked, and static memory is allocated for the model, inputs, and outputs. The main loop runs indefinitely until the power runs out or the device is shut down. In the main loop, at first the sensing data from various devices such as RGB camera, accelerometer, gyroscope, magnetometer, and IR sensors are collected and stored as inputs. The model is then invoked and the activity is classified from the input data. Finally, actions such as notifying emergency services, recording measurements, and sounding alarms are taken. A typical ML workflow contains the following stages: collecting and preparing data, designing model architectures, training the models, evaluating and optimizing models, converting models, deploying models, and performing

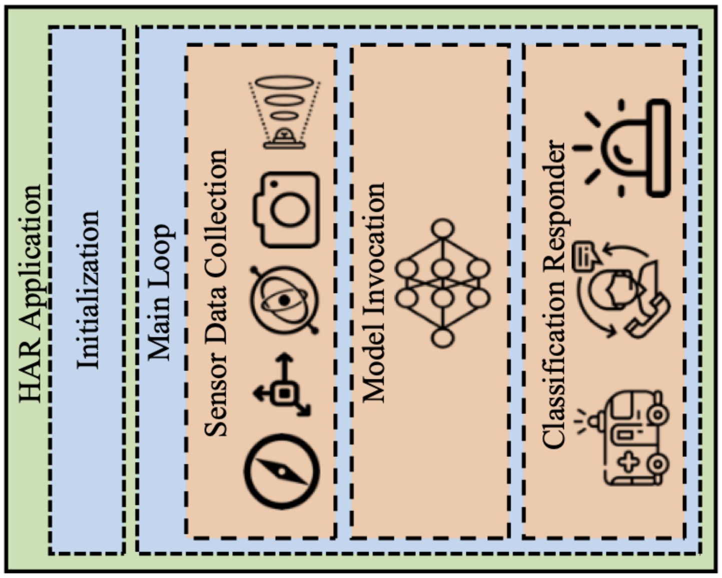


Figure 3.1: Generalization of HAR application architecture from a TinyML perspective.

inference using the models. This benchmark study is associated with multiple major stages of the typical ML workflow–from data collection and preparation to model inference. This study also overlaps with two major components of the generic HAR architecture: sensor data collection and model invocation. Each of the components used in this benchmark is described below.

3.1.1 Convolutional Neural Networks

Convolutional neural networks have emerged as one of the most prominent neural networks in the area of deep learning for a variety of important applications [101]. The powerful learning capability of deep CNN is due to the use of numerous feature extraction steps that can learn representations from input data automatically. In general, a CNN architecture consists of three types of layers: convolutional, pooling, and fully connected layers.

The convolutional layer uses numerous convolutional kernels to compute different feature maps to learn feature representations [102]. The feature value at location (p, q) in the r^{th} feature map of the s^{th} layer, $z^s_{p,q,r}$, is determined mathematically in (3.1).

$$z^s_{p,q,r} = w^{s^T}_r x^s_{p,q} + b^s_r \tag{3.1}$$

Here, the weight vector of the r^{th} filter is w^s_r, and the bias term of the s^{th} layer is b^s_r. Additionally, $x^s_{p,q}$ is the input patch centered at (p, q) of the s^{th} layer. In CNN architecture, the total number of multiplications required to convolve N kernels of size $D_k \times D_k \times M$ over an input image is as in (3.2).

$$D_k \times D_k \times M \times D_f \times D_f \times N = D_k^2 \times M \times D_f^2 \times N \tag{3.2}$$

where D_f is the dimension of the final output image.

Depthwise separable convolutions [103] instead perform the convolution operation in two steps: depthwise convolution and pointwise convolution. Depthwise convolution treats each channel in the image independently. For example, if the input image has $D_i \times D_i \times C$ dimensions, the depthwise convolution results in C number of images of dimension is $D_f \times D_f \times 1$. Pointwise convolution acts as a kernel of size $1 \times 1 \times C$ over the output images. Thus, the total number of multiplications performed by these two stages are calculated in (3.3).

$$D_k^2 \times M \times D_f^2 + M \times D_f^2 \times N = M \times D_f^2 \times (D_k^2 + N) \tag{3.3}$$

The ratio of the total number of multiplications in DSCNN and standard CNNs are given by (3.4).

$$\frac{DSCNN}{CNN} = \frac{M \times D_f^2 \times (D_k^2 + N)}{D_k^2 \times M \times D_f^2 \times N} = \frac{1}{N} + \frac{1}{D_k^2} \tag{3.4}$$

Thus, the higher number of filters and larger kernels result in lesser number of multiplications in DSCNNs compared to CNNs. DSCNNs also have lower number of learnable parameters than identical CNNs. DSCNNs can greatly improve latency, reduce memory footprint, and be more efficient than their CNN counterparts.

The activation function [104] adds nonlinearities to CNN, which are useful for detecting nonlinear features in multi-layer networks. Assume that $a(\cdot)$ is the nonlinear activation function. The activation value $a^s_{p,q,r}$ of convolutional feature $z^s_{p,q,r}$ is calculated using (3.5).

$$a^s_{p,q,r} = a(z^s_{p,q,r}) \tag{3.5}$$

The pooling layer [105] reduces the resolution of the features maps to obtain shift-invariance. Each feature map of a pooling layer is linked to its corresponding feature map of the preceding convolutional layers. Let, the pooling function is denoted as $\rho(\cdot)$, hence for each feature map $a^s_{:,:,r}$ we can calculate (3.6).

$$y^s_{p,q,r} = \rho(a^s_{m,n,r}), \forall (m,n) \in \mathbb{R}_{p,q} \tag{3.6}$$

where $\mathbb{R}_{p,q}$ is a local neighborhood around the location (p, q).

A fully connected layer, which comes after several convolutional and pooling layers, aims to conduct high-level reasoning [106]. This layer connects all neurons from the preceding stage to every neuron in the current layer to create global semantic features. Finally, the loss function is used in the output layer to obtain the optimal parameters. Let, all the parameters of CNN are denoted by θ, and the input-output relationships for data samples D are represented as $\{x^{(d)}, y^{(d)}; d \in [1, \ldots, D]\}$, where the input data of d-th is $x^{(d)}$, the corresponding target label is $y^{(d)}$, and the output of CNN is $o^{(d)}$. The loss value ($\mathcal{L}$) of CNN is determined by (3.7).

$$\mathcal{L} = \frac{1}{D} \sum_{i=1}^{D} \ell(\theta; y^{(d)}, o^{(d)}) \tag{3.7}$$

In this study, to maintain consistency across experiments, we experimented with two major types of architectures: CNNs and DSCNNs. In both cases, the input layer had 8 filters of size 3×3. The hidden layer contained 8, 16, or 32 filters of size 3×3. The base of both models contained three fully connected layers. Both the first and second base layers contained 16 neurons. The final layer had neurons equal to the number of classes of each dataset. In all layers, except the final layer Rectified Linear Unit (ReLU) was used as the activation function. The final layer used the Softmax activation function. We used Adam as the optimizer and Sparse Categorical Cross-entropy as the loss function. The original models were trained using a learning rate of 0.001. The QAT models were trained using a learning rate of 0.0001. All models were trained for 100 epochs.

3.1.2 Quantization Methods

We quantized our proposed two-stream fusion architecture using two quantization methods and tested their performance in multiple datasets. Quantization in this context is the process of converting learned model parameters such as weights, biases, and activations from higher precision to lower precision data types - reducing model size, latency, etc. in the process. Although quantization results in some information loss, in some instances, quantization can improve model performance.

Post Training Quantization

Post Training Quantization [107, 108] quantizes models without retraining. There are several ways to perform post-training quantization: Dynamic range quantization, full integer quantization, and float16 quantization. Dynamic range quantization converts 32-bit floating point values into 8-bit integers. During inference, Dynamic range quantization converts the integer values to floating point values and caches the results. This results in some speedup in inference time as well as minor performance improvements than the other PTQ methods. Full integer quantization also converts 32-bit floating point values into 8-bit integers - reducing the model size by 75% of the original model. Float16 quantization converts 32-bit floating point values into 16-bit integers - reducing the model size by 50% of the original model. PTQ methods also require a small representation dataset for optimal quantization. As PTQ aggressively reduces the precision of the parameters, some information loss occurs. For our experimentation, we used dynamic range quantization.

Quantization Aware Training

Quantization Aware Training [109, 110] requires retraining the original model for minimizing the quantization loss. QAT includes the quantization loss into overall model loss during the forward pass for each epoch - thus optimizing the total loss as the backward pass is kept the same. QAT emulates inference time quantization. During QAT, 32-bit floating point values are converted to 8-bit integer values - resulting in huge reduction in model size. QAT models almost always have lesser size than PTQ models - making QAT perfect for boards with very little Static Random Access Memory (SRAM). As QAT retrains the original model optimizing total loss, it often results in slight performance improvement over the PTQ and 32-bit original models.

3.1.3 Tiny Edge Devices

We tested the model performances in terms of power consumption and latency in three resource constrained edge devices: Arduino Nano 33 BLE Sense, Sony Spresense, and Espressif ESP32-DevKitC. Table 3.1 presents an overview of the selected devices.

Arduino Nano 33 BLE Sense

The Arduino Nano 33 BLE Sense device [111] has a 32-bit nRF52840 microprocessor with a clock speed of 64MHz. The board has 256KB SRAM and 1MB flash memory. The board is equipped with many sensors: microphone, inertial measurement unit, temperate sensor, humidity sensor, gesture sensor, pressure sensor, proximity sensor, brightness sensor, and color sensor. The board is also equipped with Bluetooh Low Energy (BLE) for communication.

Table 3.1: Overview of the boards used in experimentation

Board	**MCU/ASIC**	**Clock**	**Memory**	**Sensors**	**Radio**
Arduino Nano 33 BLE Sense	32-bit nRF52840	64MHz	1MB Flash, 256kB SRAM	M, IMU, T, H, G PS, PX, B, C	BLE
Sony Spresense	ARM $^{®}$ Cortex$^{®}$-M4F $\times$ 6	156 MHz	8MB Flash, 1.5MB SRAM	GPS	None
Espressif ESP32-DevKitC	Xtensa $^{®}$ dual-core 32-bit LX7	240 MHz	8MB Flash, 512KB SRAM	None	Wi-Fi, BLE

* M = Microphone, IMU = Inertial Measurement Unit, T = Temperature, H = Humidity, G = Gesture, PS = Pressure, PX = Proximity, B = Brightness, C = Color

Sony Spresense

The Sony Spresense device [112] has an ARM ® Cortex®-M4F microprocessor with 6 cores and a clock speed of 156MHz. The device has 1.5MB SRAM and 8MB flash memory. The board has on-board GPS but does not have any on-board radio for communication. However, additional sensors and Wi-Fi capabilities can be added by connecting an expansion board to the original board. The device also has dedicated circuitry for efficiently managing energy consumption.

Espressif ESP32-DevKitC

The Espressif ESP32-DevKitC device [113] has a Xtensa ® dual-core 32-bit LX7 microprocessor with a 240MHz clock speed. The device has 512KB of SRAM and 8MB flash memory. The device does not have any on-board sensors, but contains Wi-Fi and BLE radio systems for communication.

Among the three devices, the Espressif device has the highest clock speed, followed by the Sony device, and the Arduino device has the lowest clock speed. The Sony device has the highest SRAM, followed by Espressif, and the Arduino device has the lowest SRAM. The quantized models are stored in SRAM for faster performance. The Arduino device has the highest number of sensors. The Sony device and the Espressif device require extension boards for sensors.

3.2 Multi-resolution Fusion Architecture for HAR

In this section, we present and discuss all stages of the proposed multi-resolution fusion architecture development pipeline in the context of our human activity recognition system: dataset preprocessing, parameters of the proposed architecture, and deployment in tiny edge devices.

Figure 3.2 provides a detailed overview of our proposed system. At first, from the video sequences of human activities, we extracted the full-size greyscale images and their cropped center images. This is the input for our proposed two-stream multi-resolution fusion architecture. The two streams are symmetrical. The full-size images are input into the context stream and the cropped center images are input into the fovea stream. Due to camera bias, subjects performing activities in videos are often in the center of the frame. By inputting the center crop of the full-size images into the fovea stream, the models can extract meaningful features without distracting backgrounds or artifacts in the outskirts of the frame. The outputs of these two streams are fused through concatenation and feed into a fully connected network which outputs the activity. For example, for a 128×128 resolution still input image from a video sequence, the context stream

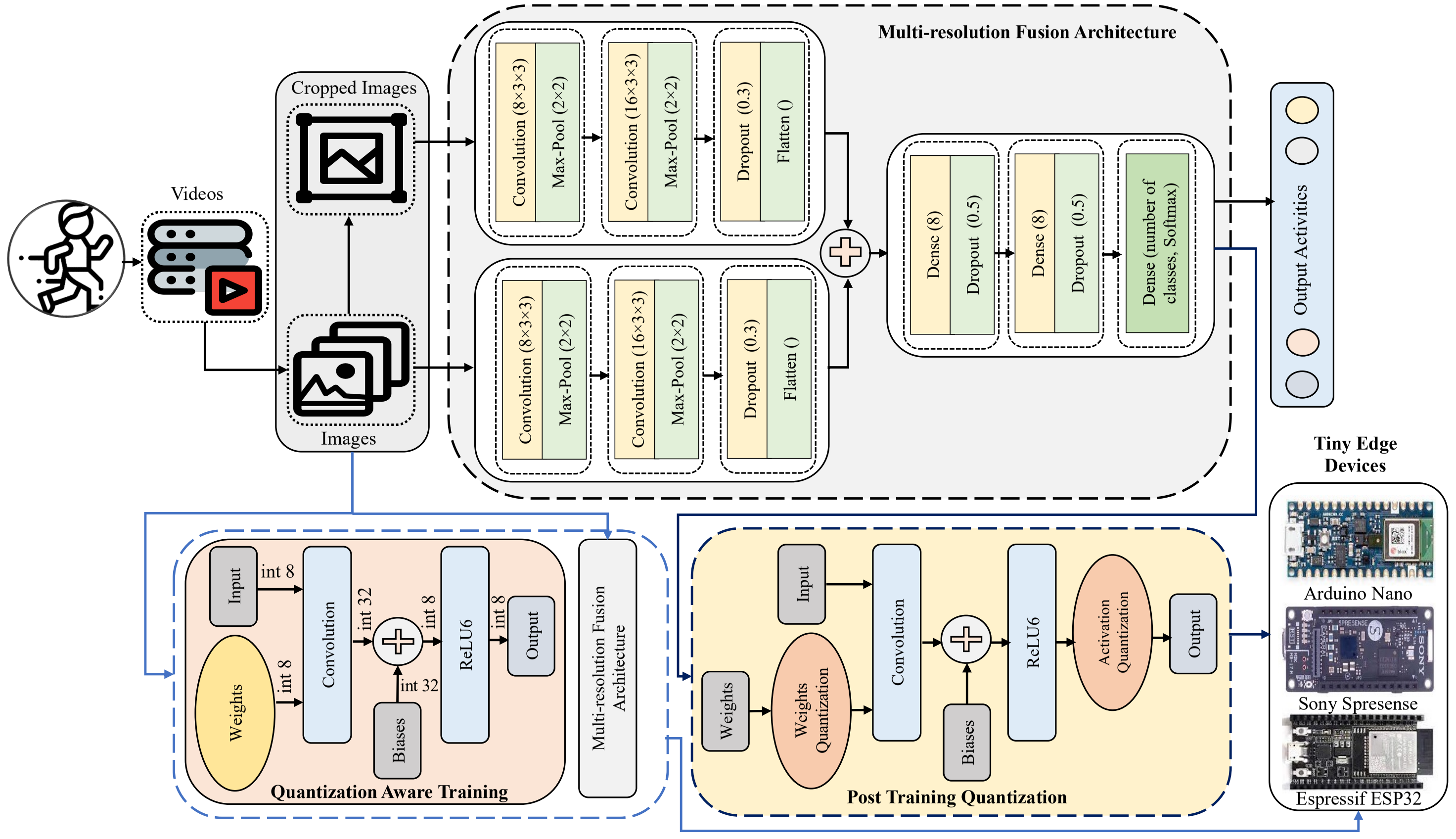


Figure 3.2: An overview of the proposed two-stream multi-resolution fusion architecture. The frames are generated from the video data and sent to the context stream to extract the significant features. The center cropped images from the frames are fed into the fovea steam for feature extraction (see top part). Later, the retrieved features are fused and sent to the fully connected layer for activity recognition. Moreover, the PTQ technique (see bottom right part) quantized the size of the developed fusion architecture and deployed the optimized model in the tiny edge devices. Further, QAT method (see bottom left part) optimized the parameters (inputs, weights, biases, and activations) of the model during the initial training and sent the optimized network to edge devices for deployment.

would receive a resized 64×64 image and the fovea stream would receive the central 32×32 pixels of the resized 64×64 image. This also reduces the dimensionality of the input by 31.2%. After the model is trained, it is quantized using PTQ where the weights and activations of the trained model are quantized into lower-precision data types and later deployed in the three tested commodity resource constrained devices: Arduino Nano 33 BLE Sense, Sony Spresense, Espressif ESP32. Another quantization technique QAT converts the input, weights, and activations of the model into lower-precision data types (int8) during the initial training. However, the biases and convolution outputs are stored in higher-precision data types (int32) for better generalization performance and reduction in important feature loss. The trained models are then deployed in the three tested edge devices.

3.2.1 Parameters of Convolutional Neural Networks

In the proposed multi-resolution fusion architecture, the context stream consists of two convolution operations with filter sizes of 8 and 16 to retrieve high-level features from the input images. After each convolution operation, a max-pooling operation is performed. At the end of each stream, a dropout layer and flatten layer is added. Dropout layers randomly disregard a given percentage of the features extracted from the previous layer. This helps the model generalize better to the training data by reducing overfitting. This might also result in a slight performance increase during testing - as the dropouts layers are not used during testing and the model gets access to all extracted features. The context stream and fovea stream are symmetrical. Both of the streams are fused using the concatenation operator. After concatenation, the fused output is input into a fully connected network. The fully connected layer has two dense layers with 8 neurons and a final dense layer with the number of neurons equal to the number of classes in the dataset. The dense layers have a dropout layer between them. The dropout layers between convolutions layers and dense layers have a dropout rate of 0.3 and 0.5.

3.2.2 TinyML Deployment

Figure 3.3 presents detailed overviews of the steps in TinyML development and deployment lifecycle: training, quantization, compilation, and deployment. After designing the proposed architecture in TensorFlow, we train the model on selected dataset. After the model is trained, we move to quantization phase. Using TFLite Micro, we perform two types of quantization: PTQ and QAT. After quantization, when we compile the model, we get the model_data.h file. This file contains the weight vectors of the quantized model as well as model metadata. We have to develop the inference logic that handles initialization, main loop and response features. We

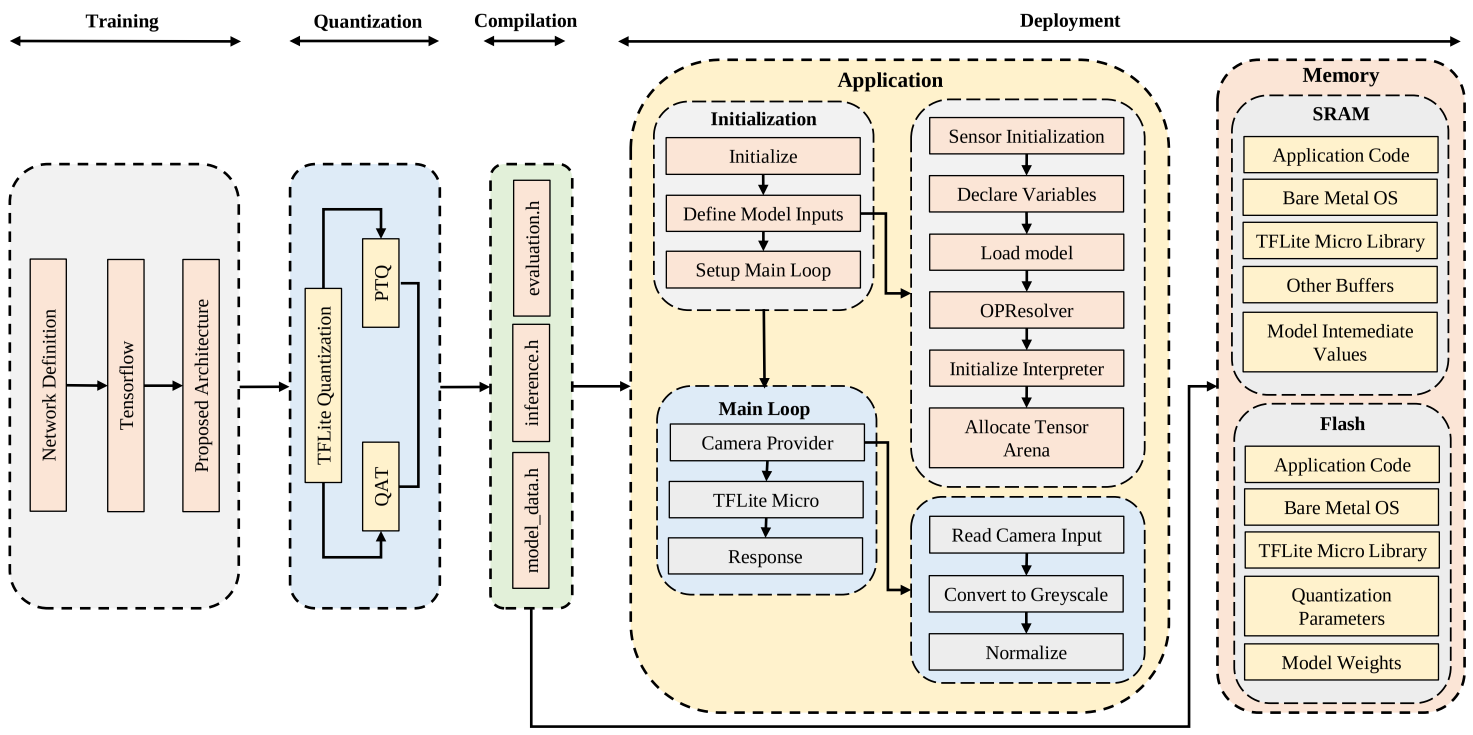


Figure 3.3: TinyML development and deployment lifecycle of our proposed multi-resolution fusion approach that includes the training of the fusion architecture using the video data, quantization of the developed architecture to reduce the size, compilation to generate necessary files compatible in low-power devices, and deployment in tiny edge devices.

also have to develop the evaluation logic to evaluate the model metrics. After we have all the files, we can deploy the models in the edge devices. In Fig. 3.3, we showed the deployment in edge devices from application and memory perspective. The standalone application has two major parts: initialization and main loop. During initialization, the device is initialized, the sensor modules are started, variables are declared, the tiny model is loaded, the OPResolver is started, the interpreter is initialized, the tensor arena is allocated, and the main infinite loop is setup. The OPResolver only loads relevant operations from the TFLite Micro library - drastically reducing space overhead. The interpreter allocates the tensor arena, interprets the operations, and runs the model. The tensor arena is a contiguous memory array holding all of the input, output and model intermediate variables. After initialization, the infinite main loop is run. The main loop reads camera input, converts the received image to greyscale, and normalizes the image. The model performs inference on the image. Based on the inference result, an appropriate action is taken. In our case, we only log the timestamp and classification result. Figure 3.3 presents the SRAM and Flash memory components for the standalone application. The SRAM component consists of application code, bare metal OS components, the TFLite Micro library, other buffers and model intermediate values. Generally, the SRAM of edge devices is limited to couple hundred KB. The flash memory component consists of the application code, bare metal OS components, TFLite Micro library, quantization parameters, and model weights. In edge devices, the flash memory is often larger in size than SRAM and limited to couple hundred KB to multiple MB. Thus, the flash memory holds the quantized weights and parameters. The SRAM holds the intermediate values from calculation, relevant TFLite Micro operations, and application code.

3.3 Summary

In this chapter, we explored the theoretical building blocks behind efficient deep learning models and their inherent differences: CNN and DSCNN, the quantization algorithms for reducing the model size without degrading model performance: QAT and PTQ, the standalone application architecture to deploy these models, as well as the commodity edge devices and their unique features: Arduino Nano 33 BLE Sense, Sony Spresense, and Espressif ESP32-DevKitC.

Chapter 4

Experimental Results and Discussions

In this chapter, we present and discuss the quantitative and qualitative results of benchmarking experiments and multi-resolution fusion architecture experiments. We tested the performance of the developed models in terms of the number of parameters, accuracy, precision, recall, F1-Score, and size in KB. We reported the performance of the developed models in terms of their inference time and power consumption in commodity edge devices. We performed and reported results of variations of the proposed architecture. Later, we discussed the tradeoffs between the model designs and provided insights into choosing the proper model and quantization strategies for various use cases.

4.1 Datasets

To perform the benchmark study on TinyML in low-power devices, we retrieved five popular benchmark datasets of human activity recognition from different well-known sources. We used two major video human activity recognition datasets to test the performance of our multi-resolution fusion models. The detailed description of each dataset is demonstrated as follows.

4.1.1 UP-Fall Detection Dataset

UP-Fall detection dataset [114] is a large multimodal human activity recognition benchmark that contains 11 activities and 3 trials per activity. The participants were involved to perform five different forms of human falls as well as six ADL. The duration of fall activities is 10 seconds, while the duration of ADL is variable. The data has been collected from a laboratory environment using wearable sensors, ambient sensors, and vision devices. The dataset contained two types of data: time-series data collected from wearable sensors, and ambient sensors; video/image data collected from camera devices. Additionally, the vision data has two views including frontal views (camera 2) and lateral views (camera 1). For this research, we have selected only the vision data considering the frontal views. The image data were resized to three different input sizes: $16 \times 16 \times 3, 32 \times 32 \times 3$, and $64 \times 64 \times 3$ for experimentation.

4.1.2 Fall Detection Dataset

Fall Detection Dataset [115] has been collected from 5 different controlled rooms through an uncalibrated Kinect sensor in the shape of raw RGB and depth image with a size of 640×480. Five volunteers (2 males, and 3 females) performed five different activities in the controlled environments. The dataset contains 21499 images in total. For the research purpose, the dataset is divided into three sets: training (15800 images), validation (3199 images), and testing (2500 images). The total number of images are in sequence, but none of them are repeated, and all of the sets have original and horizontally flipped samples incorporated in sequence to maximize the number of samples in each set. Similar to the UP-Fall detection dataset, the input images were resized to $16 \times 16 \times 3, 32 \times 32 \times 3$, and $64 \times 64 \times 3$ input sizes for experimentation.

4.1.3 PAMAP2

PAMAP2 dataset [116] is an activity monitoring dataset that contains 18 different household activities which were collected from 9 different volunteers (8 men, and 1 woman) through the use of a heart rate monitor and three Inertial Measurement Unit (IMU)s such as accelerometers, gyroscopes, and magnetometers. The IMUs are positioned at the hand, chest, and ankle whereas the heart rate monitor is placed at the chest for data collection purposes. The sampling frequency for this dataset is 100 Hz. In this dataset, 10 seconds of data at the start and the end of each labeled activity event is eliminated to prevent dealing with possible transient activities. We processed 120 consequent data sequences and resized into $20 \times 20 \times 3$ input size for experimentation.

4.1.4 UCI-HAR

UCI-HAR dataset [117] has been frequently utilized in human activity recognition research since its publication. This dataset is also known as the behavior recognition dataset and was collected from 30 subjects considering six daily activities through the use of accelerometers and gyroscope operating at 50 Hz in a supervised scenario. The three-component values of the accelerometer and gyroscope are acquired independently, and the data dimension is 561. The data collection processes have been video-recorded to label the data manually. The activity recognition procedure starts with the collection of sensor data, which are then pre-processed with noise filters and recorded in 2.56 seconds of fixed-width sliding windows with a 50% overlap. The retrieved data has been randomly partitioned into training set with 70% subjects, and testing set with 30% subjects. As every sample of the UCI-HAR dataset contains pre-processed features extracted from multiple sequences of activity, the input was resized to $17 \times 11 \times 3$ for experimentation.

4.1.5 WISDM

Wireless Sensor Data Mining (WISDM) dataset [118] has been collected from 36 subjects in a strictly controlled environment using tri-axial accelerometer of smartphone. Each of the volunteers kept the smartphone in their front leg trouser pockets, and was requested to perform six different types of low-level activities. The longitudinal, lateral, and forward activity signals are collected through an accelerometer at a sampling frequency of 20 Hz. For this dataset, the length of the sliding window is 10 seconds and the overlap rate is 90%. The dataset contains 1098209 samples overall in the form of time-series and the nature of the data is imbalanced. We processed 100 input data sequences and resized them into $10 \times 10 \times 3$ input size for experimentation.

4.1.6 KTH Human Activity Dataset

The KTH human activity dataset [119] contains a total of 2391 video sequences of six human actions performed by 25 subjects in 4 unique scenarios: indoor, outdoor, outdoor with different clothes, outdoor with scale variation. We used 4 human actions from the KTH dataset: boxing ($K1$), handclapping ($K2$), handwaving ($K3$), and walking ($K4$). We did not use the running and jogging activities as the still images from these activities are indistinguishable from the walking activity without temporal information. The original video sequences had a spatial resolution of 160×120 pixels and were captured over homogeneous backgrounds with a static camera at 25fps frame rate. We extracted the single images from the videos sequences, resized the greyscale images into 128×128, 64×64, and 32×32, and cropped the center images into resolution: 64×64, 32×32, and 16×16. We normalized all the images by dividing the pixel intensities of each image by 255.0.

4.1.7 UCF11 YouTube Action Dataset

The UCF11 YouTube Action Dataset [120] contains 11 unique activities: basketball ($U1$), biking ($U2$), diving ($U3$), golf_swing ($U4$), horse_riding ($U5$), soccer_juggling ($U6$), swinging ($U7$), tennis_swing ($U8$), trampoline_jumping ($U9$), volleyball_spiking ($U10$), and walking ($U11$). We trained and tested all 11 activities of this dataset. The dataset is very challenging for recognition tasks due to huge variations in camera motion in the sequences, varying illumination conditions, changing viewpoints, cluttered backgrounds, different object scales, and differences in object pose and appearance. The dataset contains a total of 1168 video sequences. As the dataset was collected from YouTube, the videos are from different subjects and have different resolutions. For preprocessing, the single-frame images are extracted from the video sequences and converted to greyscale. The greyscale images are then resized into 128×128, 64×64, and 32×32, and their center portions are cropped into resolution: 64×64, 32×32, and 16×16. The resized and cropped images are normalized by dividing the pixel intensities of each image by 255.0.

4.2 Performance Evaluation

To demonstrate the effectiveness and feasibility of the benchmark models and the proposed fusion architectures, we utilize two different classes of performance metrics: model performance metrics and device performance metrics. Model performance metrics track the performance of the developed models while device performance metrics track their on-device performance in deployment scenarios.

4.2.1 Model Performance

We measure the efficiency both of the benchmark models and the proposed fusion architecture in terms of four widely used statistical metrics: accuracy, precision, recall, and F1-Score. Here, TP, FP, TN, and FN represent True Positive, False Positive, True Negative, and False Negative, respectively. Due to the multiclass nature of the datasets, we used weighted average to calculate the performance metrics. Weighted average considers the frequency of the class labels during the calculation of the metrics.The four metrics are defined as follows.

- **Accuracy**: Accuracy is the ratio of correctly recognized samples in all data samples that represent the rate of recognition of human activities.

$$Accuracy = \frac{1}{N}\sum_{k=1}^{N}\frac{TP_k + TN_k}{TP_k + TN_k + FP_k + FN_k} \tag{4.1}$$

- **Precision**: Precision is the ratio of real positive samples to the total number of recognized positive data instances.

$$Precision = \frac{1}{N}\sum_{k=1}^{N}\frac{TP_k}{TP_k + FP_k} \tag{4.2}$$

- **Recall**: Recall represents the proportion of correctly recognized positive samples to the total number of real positive data instances.

$$Recall = \frac{1}{N}\sum_{k=1}^{N}\frac{TP_k}{TP_k + FN_k} \tag{4.3}$$

- **F1-Score**: F1-Score is a measure to evaluate the performance of multi-classification architectures that is represented by the weighted average of recall and precision which facilitates to give more insights into human activity recognition tasks.

$$F1 - Score = \frac{1}{N}\sum_{k=1}^{N}\frac{2 * TP_k}{2 * TP_k + FP_k + FN_k} \tag{4.4}$$

4.2.2 Device Performance

To evaluate the performance of our proposed models, we developed standalone applications utilizing these models and deployed them in commercially available edge devices. We used inference time and power consumption measurements to track their performance.

- **Inference Time**: Inference time indicates the total time taken to perform a forward propagation in a model. In our case, the standalone application logs two timestamps: one before the model is invoked and one after classification is performed. The difference between these two timestamps is the inference time for that sample. We calculated the average inference time from all test samples.

- **Power Consumption**: Power consumption represents the average electric current drawn while running the standalone application in mA reported by the Power Profiler Kit.

4.3 Experimental Setup

The computer used for these experiments had the following specifications: CPU: Intel® Core™ i7-6700K 8-core CPU @ 4.00GHz, GPU: GTX 1080 with 8 GB GDDR5 memory, RAM: 32 GB

DDR4. The computer was running Ubuntu 22.04 LTS with Python(v3.10), TensorFlow(v2.8.0), and Tensorflow Model Optimization(v0.7.2) libraries. Nordic® Power Profiler Kit II was used to measure the power consumption of the devices deploying the models.

4.4 Benchmarking HAR

We elaborate on the test results on the selected datasets as well as the inference time and power consumption in selected devices.

4.4.1 Results Analysis

In the UP-Fall detection dataset and FDD dataset, the data was normalized. In the PAMAP2, UCI-HAR, and WISDM datasets, the data were standardized. We performed 80% − 20% train-test split on the UP-Fall, PAMAP2, and WISDM datasets. FDD and UCI-HAR datasets provided separate training and testing sets.

In all subsequent Tables in this subsection, the gray rows indicate DSCNN model and the white rows indicate CNN model. We will discuss and compare the model performance between QAT and PTQ versions only in the following stages as our focus is on TinyML. Table 4.1 presents the experimental results for the UP-Fall detection dataset. For the UP-Fall detection dataset, for input size $16 \times 16 \times 3$, the highest levels of accuracy, precision, recall and F1-Score of 89.4%, 89.4%, 89.4%, and 89.3%, respectively, was achieved when QAT and 16 hidden layers in CNN network was used. For $32 \times 32 \times 3$ input size, the highest accuracy, precision, recall, and F1-Score of 92.2%, 92.6%, 92.2%, and 92.2%, respectively, was achieved when QAT and 16 hidden layers in CNN was used. For $64 \times 64 \times 3$ input size, the highest metrics were achieved when QAT and 8 hidden layers in CNN was used. The highest accuracy, precision, recall, and F1-Score are 95.5%, 95.6%, 95.5%, and 95.4%, respectively.

Table 4.2 presents the experimental results for the FDD dataset. For the FDD datasets, for input size of $16 \times 16 \times 3$, the highest accuracy, precision, recall, and F1-Score of 79.3%, 78.3%, 79.3%, and 78.4%, respectively, was achieved when QAT and 32 hidden layers in CNN was used. For $32 \times 32 \times 3$ input size, the highest accuracy, recall, and F1-Score was achieved when QAT and 32 hidden layers in CNN were used. The highest accuracy, recall, and F1-Score were 81.7%, 81.7%, and 81.2%, respectively. For $64 \times 64 \times 3$ input size, the highest accuracy, recall, and F1-Score of 86.9%, 86.9%, and 86.5%, respectively, was achieved when PTQ and 32 hidden layers in DSCNN was used.

Table 4.1: Experimental results for the UP-FALL detection dataset

IS	H	PM	Performance (%)					QAT(%)					PTQ(%)				
			A	P	R	F1	S	A	P	R	F1	S	A	P	R	F1	S
	8	1795	83.4	83.8	83.4	83.5	186.7	84.2	85.1	84.2	84.4	7.1	83.4	83.8	83.4	83.5	10.3
	16	2891	88.8	89.0	88.8	88.8	201.2	89.4	89.4	89.4	89.3	8.4	88.8	89.0	88.8	88.8	14.6
	32	5083	87.4	87.5	87.4	87.3	227.8	86.3	87.7	86.3	86.8	11.0	87.4	87.5	87.4	87.3	23.2
16	8	1190	82.2	82.2	82.2	82.0	210.7	83.0	83.0	83.0	82.9	8.1	82.2	82.2	82.2	82.0	9.1
	16	1774	85.1	85.4	85.1	85.2	218.5	54.0	48.2	54.0	48.4	9.0	85.1	85.4	85.1	85.2	11.5
	32	2942	86.4	86.7	86.4	86.3	232.6	87.3	87.3	87.3	87.2	10.5	86.4	86.7	86.4	86.3	16.0
	8	5891	90.9	91.1	90.9	90.8	238.6	91.7	91.9	91.7	91.7	11.2	90.9	91.1	90.9	90.8	26.4
	16	11083	91.9	92.2	91.9	91.8	300.9	92.2	92.6	92.2	92.2	16.5	91.9	92.2	91.9	91.8	46.6
	32	21467	18.3	03.3	18.3	05.0	425.5	18.5	03.4	18.5	05.7	27.0	18.3	03.3	18.3	05.6	87.2
32	8	5286	90.8	91.4	90.8	90.8	260.8	92.0	92.0	92.0	91.9	12.2	90.8	91.4	90.8	90.8	25.2
	16	9966	88.2	88.1	88.2	88.0	317.2	87.5	88.4	87.5	87.8	17.0	88.2	88.1	88.2	88.0	43.5
	32	19326	18.5	03.4	18.5	05.7	430.1	18.5	03.4	18.5	05.7	26.6	18.5	03.4	18.5	05.7	80.1
	8	26371	95.0	95.1	95.0	94.9	481.8	95.5	95.6	95.5	95.4	31.1	95.0	95.1	95.0	94.9	106.3
	16	52043	91.6	92.3	91.6	91.8	789.8	92.5	92.5	92.5	92.4	56.3	91.6	92.3	91.6	91.8	206.5
	32	103387	18.3	03.3	18.3	05.7	1400.0	18.5	03.4	18.5	05.7	106.9	18.3	03.3	18.3	05.7	407.1
64	8	25766	91.4	91.5	91.4	91.2	504.2	92.5	92.5	92.5	92.3	32.0	91.4	91.5	91.4	91.2	105.1
	16	50926	93.6	93.8	93.6	93.6	806.1	93.7	94.0	93.7	93.6	56.8	93.6	93.8	93.6	93.6	203.4
	32	101246	18.5	03.4	18.5	05.7	1400.0	18.5	03.4	18.5	05.7	106.4	18.5	03.4	18.5	05.7	399.9

* IS = Input Shape, H = Hidden Layer, PM = Parameters, A = Accuracy, P = Precision, R = Recall, F1 = F1-Score, S = Size in KB.

Table 4.2: Experimental results for the FDD dataset

IS	H	PM	Performance (%)					QAT(%)					PTQ(%)				
			A	P	R	F1	S	A	P	R	F1	S	A	P	R	F1	S
	8	1710	69.3	69.8	69.3	68.1	185.5	71.1	70.0	71.1	69.5	7.0	69.3	69.0	69.3	68.1	10.0
	16	2806	69.2	69.1	69.2	68.5	199.9	72.9	72.3	72.9	71.8	8.3	69.2	69.1	69.2	68.5	14.3
	32	4998	77.1	76.6	77.1	75.0	226.3	79.3	78.3	79.3	78.4	10.9	77.1	76.6	77.1	75.0	22.8
16	8	1105	59.4	59.7	59.4	58.6	209.2	59.8	60.5	59.8	59.4	8.0	59.4	59.7	59.4	58.6	8.8
	16	1689	64.8	64.8	64.8	62.8	217.0	63.5	60.9	63.5	60.5	8.8	64.8	64.8	64.8	62.8	11.1
	32	2857	65.2	65.7	65.2	64.3	231.4	66.9	66.2	66.9	65.4	10.4	65.2	65.7	65.2	64.3	15.7
	8	5806	77.0	77.0	77.0	75.9	237.3	75.7	75.1	75.7	74.5	11.1	77.0	77.0	77.0	75.9	26.1
	16	10998	79.1	79.7	79.1	78.6	299.6	81.4	81.5	81.4	81.1	16.4	79.1	79.7	79.1	78.6	46.3
	32	21382	78.8	79.9	78.8	78.3	424.3	81.7	81.2	81.7	81.2	26.9	78.8	79.9	78.8	78.3	86.9
32	8	5201	50.5	53.8	50.5	47.1	259.5	56.9	55.7	56.9	54.9	12.1	50.5	53.8	50.5	47.1	24.8
	16	9881	71.9	73.1	71.9	71.2	315.9	70.0	69.7	70.0	68.0	16.9	71.9	73.1	71.9	71.2	43.2
	32	19241	77.2	77.7	77.2	76.0	428.8	73.8	73.0	73.8	71.8	26.5	77.2	77.7	77.2	76.0	79.8
	8	26286	82.7	83.4	82.7	82.2	483.3	82.6	82.4	82.6	82.2	31.2	82.7	83.4	82.7	82.2	106.1
	16	51958	86.5	87.4	86.5	86.1	791.3	86.6	86.6	86.6	86.3	56.4	86.5	87.4	86.5	86.1	206.3
	32	103302	79.8	80.1	79.8	77.9	1400.0	78.2	78.0	78.2	75.8	107.0	79.8	80.1	79.8	77.9	406.9
64	8	25681	76.3	75.1	76.3	74.8	506.1	72.2	69.4	72.2	68.7	32.1	76.3	75.1	76.3	74.8	104.9
	16	50841	74.2	75.3	74.2	73.9	808.0	72.2	72.2	72.2	70.6	56.9	74.2	75.3	74.2	73.9	203.2
	32	101161	86.9	86.6	86.9	86.5	1400.0	79.0	78.2	79.0	78.2	106.5	86.9	86.6	86.9	86.5	399.8

* IS = Input Shape, H = Hidden Layer, PM = Parameters, A = Accuracy, P = Precision, R = Recall, F1 = F1-Score, S = Size in KB.

Table 4.3 provides the experimental results for the time-series datasets: PAMAP2, UCI-HAR, and WISDM. For the PAMAP2 dataset, the highest levels of accuracy, recall and F1-Score of 91.2% was achieved when PTQ and 32 hidden layers in CNN were used. However, the performance of the QAT and PTQ models were very similar. For the UCI-HAR dataset, the highest metrics of accuracy, recall, and F1-Score were 78.2%, 78.2%, and 78.3%, respectively. These metrics were achieved when PTQ and 32 hidden layers in CNN was used. For the WISDM dataset, the highest accuracy, recall, and F1-Score of 87.8%, 87.8%, and 87.3%, respectively, was achieved when QAT and 32 hidden layers in CNN was used.

Table 4.4 presents the percentile delta between the performance metrics of QAT and PTQ models for each dataset. The delta for each performance metric was calculated using (4.5). Here, V is the property, we are interested in. $\overline{V_{QAT}}$ and $\overline{V_{PTQ}}$ are average values of the property across a specific input size for the QAT and PTQ model. The delta was calculated by averaging all the values for a specific input size. A negative delta in our case signifies that the value from PTQ model is larger than the value form QAT model and vice versa. A positive delta value for A, P, R, and $F1$ indicate the QAT models performed better on average than PTQ models. A negative delta value of S indicate that PTQ models were on average larger in size than QAT models. From Table 4.4, it is evident that the PTQ models are consistently larger in size than their counterpart QAT models. In the majority of cases, the QAT models outperformed the PTQ models.

$$\Delta V = \overline{V_{QAT}} - \overline{V_{PTQ}} \tag{4.5}$$

In all our experiments, the CNN models achieved higher levels of accuracy, precision, recall and F1-Score in general compared to their DSCNN counterparts. While DSCNN resulted in lower number of total parameters in all cases, DSCNN models also used marginally more storage space in both QAT and PTQ use cases. DSCNN models also had marginally smaller latency and power consumption. This is due to the fact that, theoretically DSCNN models use lesser Multiply-accumulate (MAC) operations than their CNN counterparts. This results in lesses number of computations for CPUs in resource constrained devices. The PAMAP2 dataset has significantly more inference latency and power draw due to the significantly larger input size. The QAT models outperformed the PTQ models in most cases. However, the PTQ models consistently provided performance metrics similar to the originally trained model, whereas, sometimes the QAT model would under-perform than the original model. The PTQ models also require representative datasets which the QAT models do not need. PTQ models also required more storage space than the QAT models.

Table 4.3: Experimental results for all time-series datasets

IS	H	PM	Performance (%)					QAT(%)					PTQ(%)				
			A	P	R	F1	S	A	P	R	F1	S	A	P	R	F1	S
	8	2452	80.4	79.6	80.4	79.4	205.4	79.8	78.5	79.8	78.6	8.03	80.4	79.6	80.4	79.4	13.0
	16	4188	90.5	90.4	90.5	90.4	228.0	90.7	90.5	90.7	90.6	10.0	90.5	90.4	90.5	90.4	19.8
PAMAP2	32	7660	91.2	91.1	91.2	91.2	269.7	91.1	91.0	91.1	91.0	13.8	91.2	91.1	91.2	91.2	33.4
20×20	8	1847	82.0	81.8	82.0	81.4	228.5	82.1	82.0	82.1	81.7	9.07	82.0	81.8	82.0	81.4	11.8
	16	3071	82.4	82.1	82.4	81.9	244.2	83.1	82.6	83.1	82.6	10.5	82.4	82.1	82.4	81.9	16.6
	32	5519	85.5	85.2	85.5	85.1	273.7	85.4	84.7	85.4	84.7	13.3	85.5	85.2	85.5	85.1	26.2
	8	1454	69.7	71.6	69.7	69.6	182.4	71.2	71.9	71.2	71.3	6.7	69.7	71.6	69.7	69.6	9.0
UCI	16	2294	73.9	74.8	73.9	74.1	193.8	75.2	75.4	75.2	75.3	7.8	73.9	74.8	73.9	74.1	12.3
HAR	32	3974	78.2	78.7	78.2	78.3	214.0	78.0	78.4	78.0	78.1	9.9	78.2	78.7	78.2	78.3	18.8
17×11	8	849	63.4	64.0	63.4	63.0	206.3	64.9	64.6	64.9	64.6	7.8	63.4	64.0	63.4	63.0	7.8
	16	1177	69.9	70.1	69.9	69.8	211.1	68.9	69.0	68.9	68.8	8.3	69.9	70.1	69.9	69.8	9.1
	32	1833	57.6	59.2	57.6	57.9	219.0	56.7	56.9	56.7	56.7	9.4	57.6	59.2	57.6	57.9	11.7
	8	1326	84.2	82.7	84.2	83.2	180.9	85.6	84.3	85.6	84.4	6.6	84.2	82.7	84.2	83.2	8.5
	16	2038	86.7	85.8	86.7	85.9	190.9	87.1	86.7	87.1	86.7	7.6	86.7	85.8	86.7	85.9	11.3
WISDM	32	3462	87.4	87.7	87.4	87.2	208.0	87.8	87.4	87.8	87.3	9.4	87.4	87.7	87.4	87.2	16.8
10×10	8	721	82.0	79.7	82.0	79.9	204.8	81.1	77.1	81.1	78.6	7.7	82.0	79.7	82.0	79.9	7.3
	16	921	84.3	81.4	84.3	82.4	208.0	84.3	82.0	84.3	82.9	8.1	84.3	81.4	84.3	82.4	8.1
	32	1321	84.8	83.5	84.8	83.3	212.9	85.7	84.5	85.7	84.8	8.9	84.8	83.5	84.8	83.3	9.7

* IS = Input Shape, H = Hidden Layer, PM = Parameters, A = Accuracy, P = Precision, R = Recall, F1 = F1-Score, S = Size in KB.

Table 4.4: Delta of performance metrics between QAT and PTQ models.

Dataset	Input Size	ΔA	ΔP	ΔR	ΔF1	ΔS
UP-Fall	16×16	0.10	0.63	0.10	0.30	-7.20
		-9.80	-11.9	-9.80	-11.6	-3.00
	32×32	0.43	0.43	0.43	-3.8	-35.1
		0.16	0.30	4.43	-3.96	-31.0
	64×64	0.53	0.26	0.53	0.36	-175
		0.4	0.4	0.4	0.3	-171
FDD	16×16	-0.76	1.96	2.56	2.7	-6.96
		0.06	-0.86	0.26	-0.13	-2.8
	32×32	1.3	0.4	1.3	1.3	-34.9
		0.36	-2.06	0.36	0.13	-30.7
	64×64	-0.53	-1.30	-0.53	-0.63	-174
		-4.66	-5.73	-4.66	-5.90	-170
PAMAP2	20×20	-0.20	-0.36	-0.16	-0.26	12.6
		0.23	0.06	0.23	0.20	19.9
UCI-HAR	17×11	1.16	0.20	0.86	0.90	-5.23
		-0.13	-0.93	-0.13	-0.20	-1.03
WISDM	10×10	0.73	0.73	0.73	0.70	-4.33
		0.01	-0.33	0.01	0.23	-0.13

4.4.2 Inferrence Time and Power Consumption

Fig. 4.1 presents the average inference time in milliseconds (ms) for each device and dataset. Generally, the Espressif ESP-32 device has lower inference time than the other devices for the same dataset. The DSCNN models also have generally lower inference time than the CNN models. Fig. 4.2 presents the average power consumption for each device and dataset. The Espressif ESP-32 device had higher power consumption than the other devices due to the higher clock speed of the processor. The DSCNN models also had marginally lower power consumption than CNN models due to the decreased number of required computations. The power consumption and inference time can also be significantly reduced if grayscale images are used instead of color images. As the purpose is activity classification, converting color images to grayscale images should have marginal effect on model performance while greatly affecting inference time and

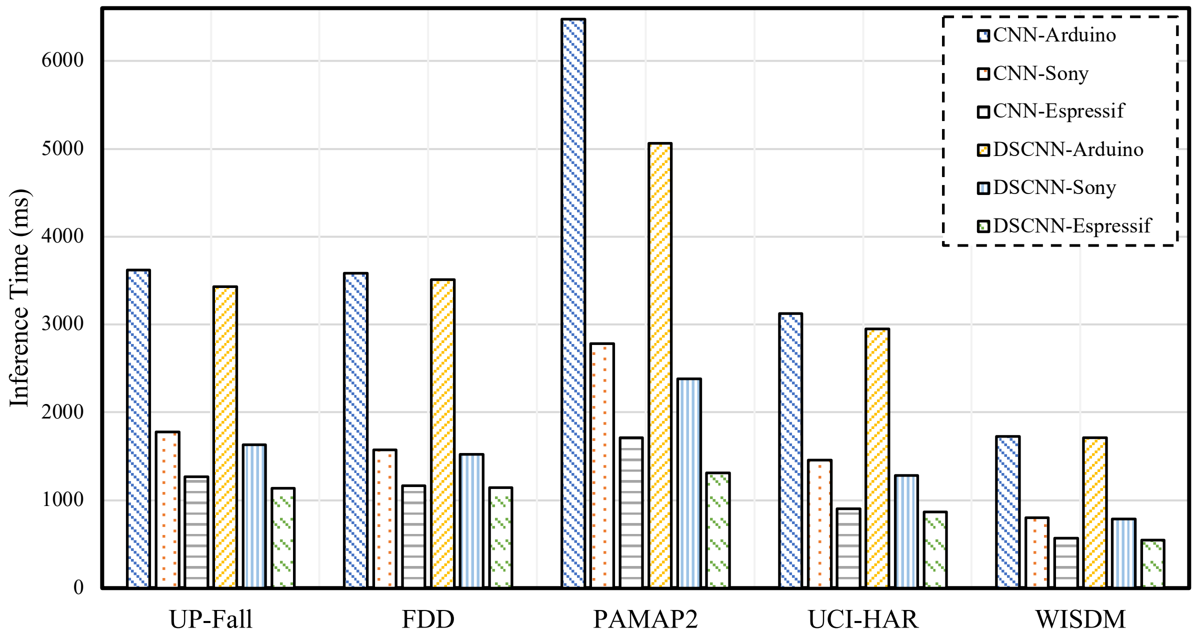


Figure 4.1: Inference time for the datasets in the tested boards.

power consumption.

In terms of inference time, the device with the higher clock speed generally resulted in lower inference time. However, the higher clock speed resulted in more power consumption than lower clock speed counterparts. The tested Sony Spresense device has built in power management circuitry and multi-core processing support, which resulted in generally low inference time and low power consumption.

4.5 Multi-resolution Fusion Architecture

We present the test results of our proposed two-stream multi-resolution fusion architecture on the KTH and UCF11 dataset, the performance after quantization using QAT and PTQ, ablation studies, and deployment performance.

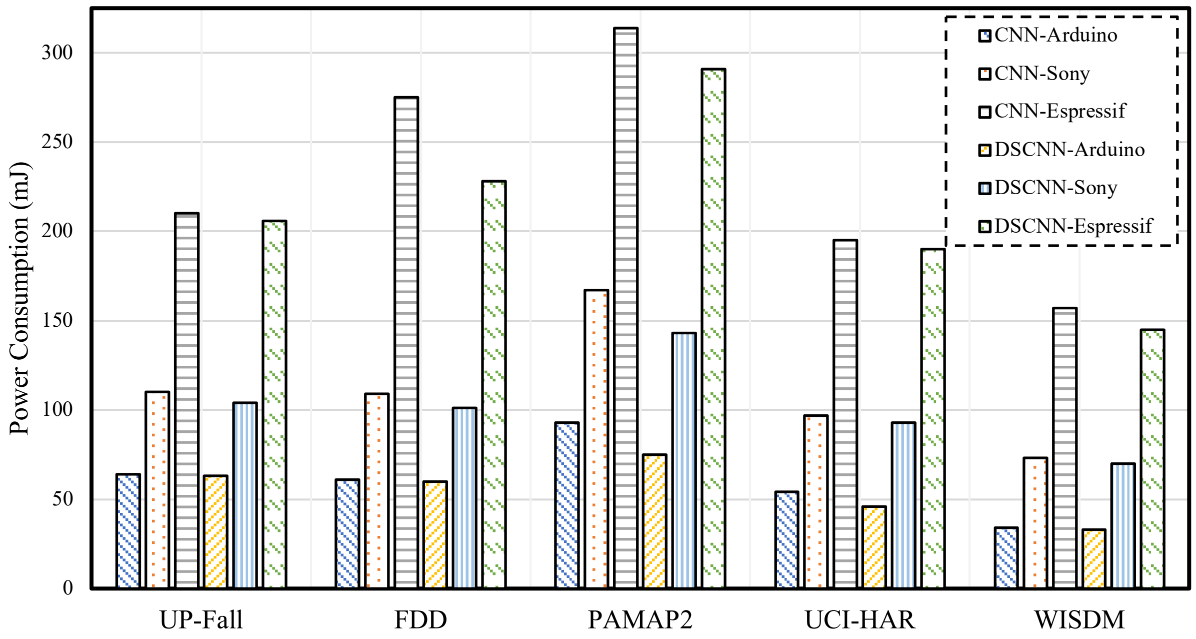
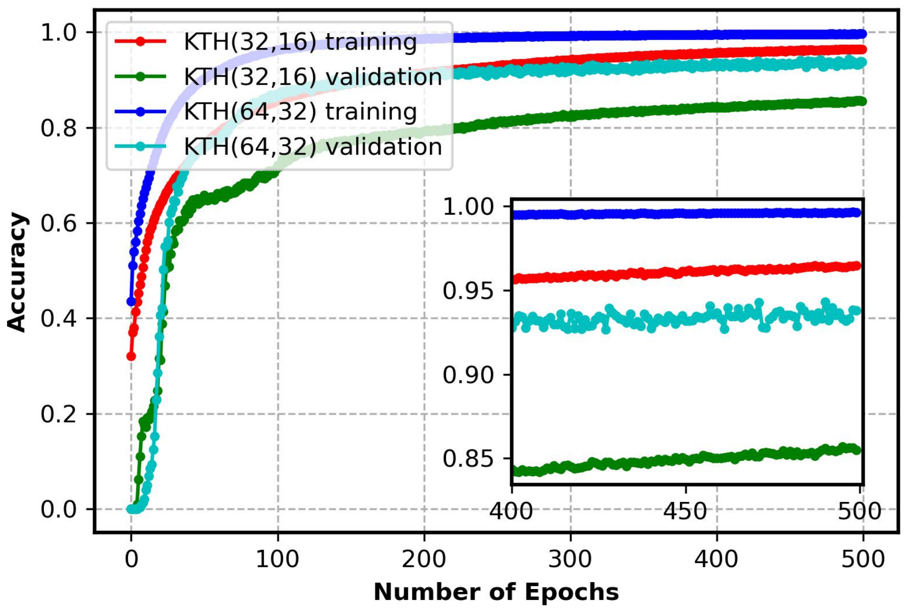


Figure 4.2: Power consumption for the datasets in the tested boards.

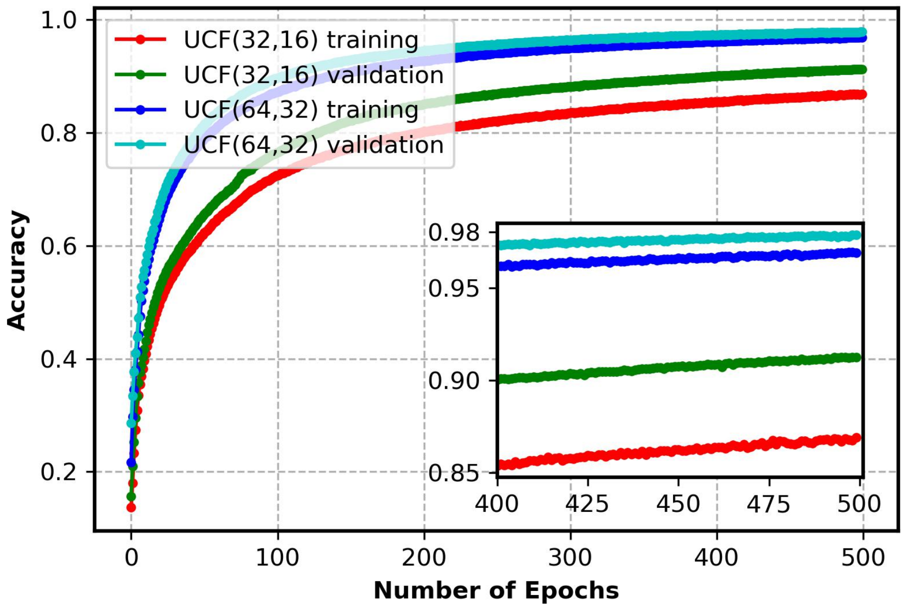
4.5.1 Results Analysis

We used Adam optimizer and sparse categorical cross-entropy loss to train the model. The learning rate for training the model is set to 0.0001. The models for each experiment were trained for 500 epochs. We used 5-fold cross-validation to ensure the model performances were robust to training data. The average metrics of the cross-validation run were presented as the model performance in the subsequent tables. In all subsequent tables, white rows represent the results on KTH dataset and gray rows represent the result on UCF11 dataset. We represent a model in all subsequent tables using the format: $DATASET(x, y)$. $DATASET(x, y)$ indicates that the context stream takes an input of size $x \times x$ and the fovea stream takes an input of size $y \times y$.

Figure 4.3 illustrates the accuracy with respect to epoch while training the proposed fusion architecture on KTH and UCF11 datasets. The KTH (64,32) model reached peak training and validation accuracy around 200 epochs. The accuracy curves for training and validation are similar to one another without any major changes, indicating that the model is generalizing well to the dataset. The KTH (32,16) model reached similar accuracy to KTH (64,32) model while training but reached lower accuracy while validating. The UCF (64,32) model reached higher training and validation accuracy than the UCF (32,16) model. This result is mirrored in Table 4.7.

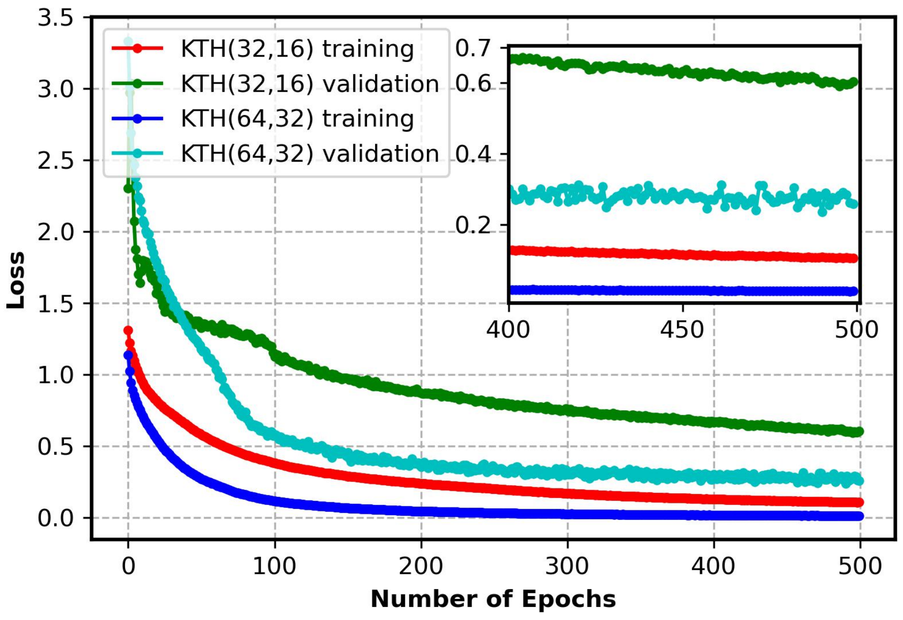


(a)

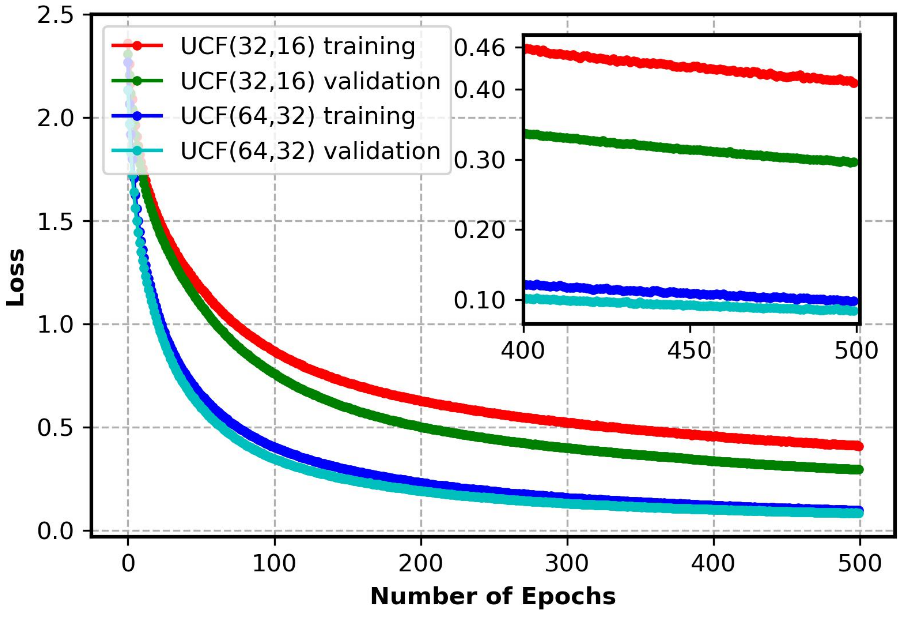


(b)

Figure 4.3: Performance measures (accuracy curves) of the developed two-stream mutti-resolution fusion approach. (a) KTH (b) UCF11.

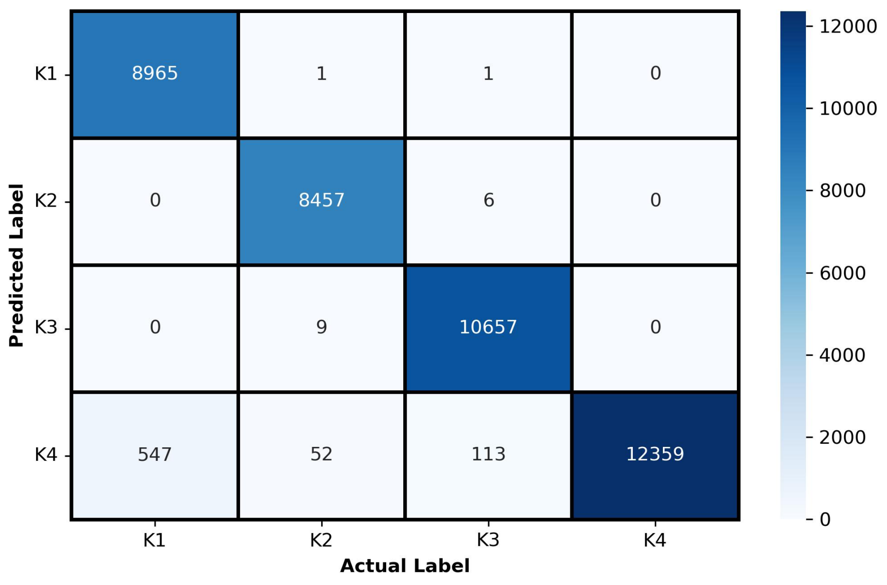


(a)

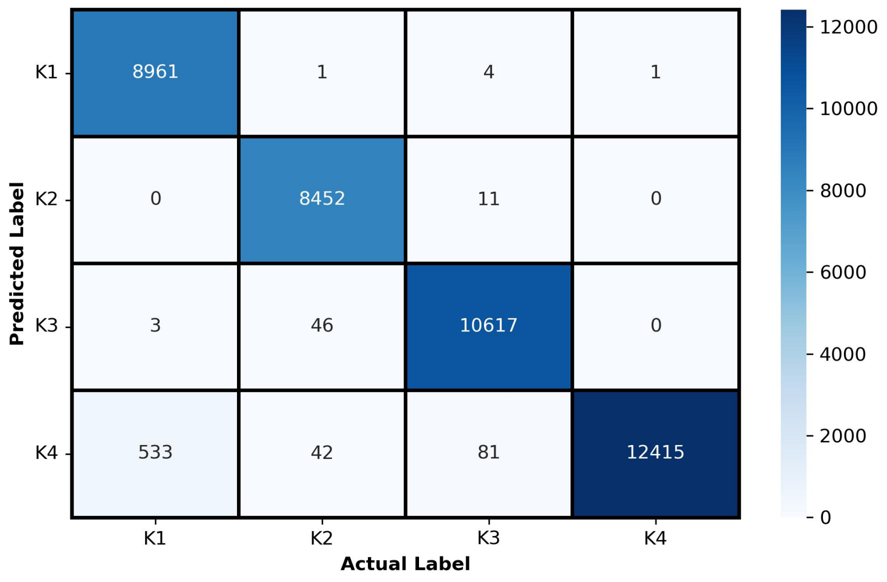


(b)

Figure 4.4: Performance measures (loss curves) of the developed two-stream mutti-resolution fusion approach. (a) KTH (b) UCF11.

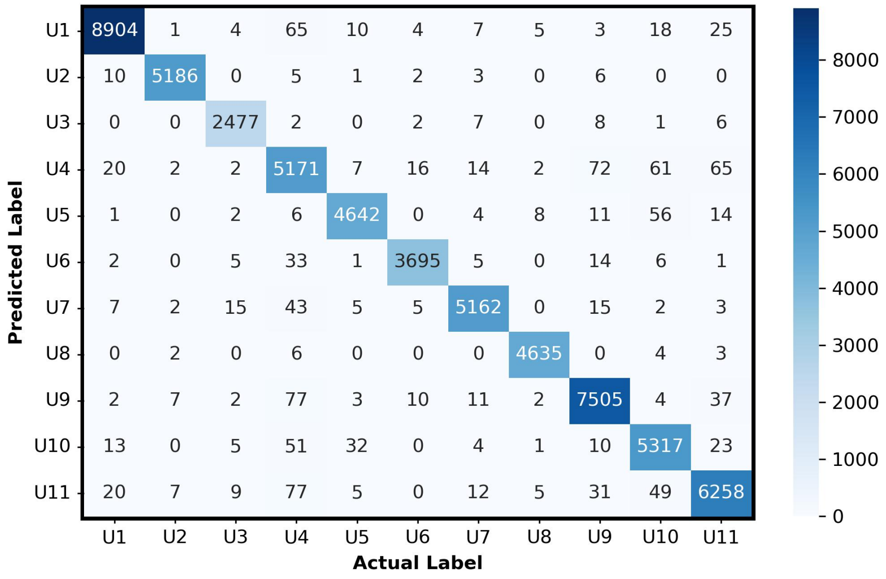


(a)



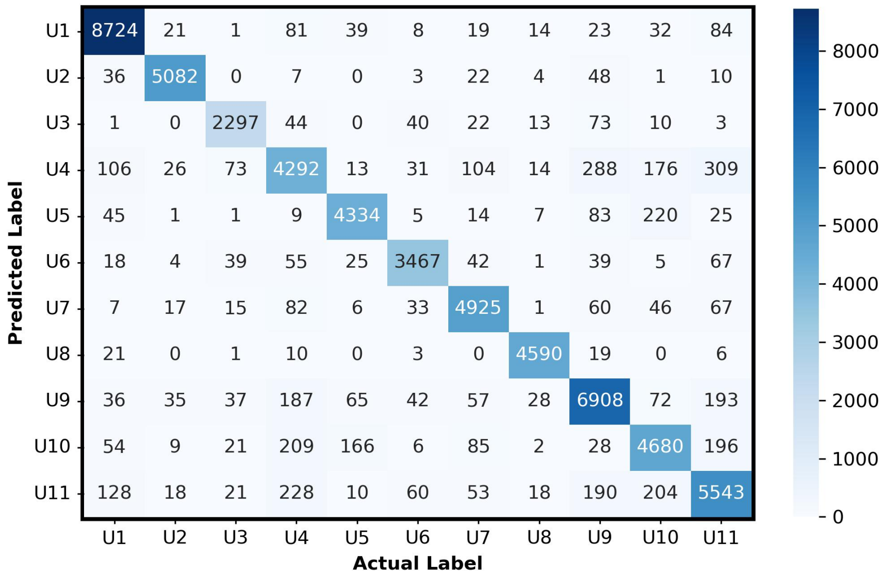
(b)

Figure 4.5: Confusion matrix of the two-stream multi-resolution fusion architecture for KTH dataset. (a) KTH (64,32) and (b) KTH (32,16).



	U1	U2	U3	U4	U5	U6	U7	U8	U9	U10	U11
U1	8904	1	4	65	10	4	7	5	3	18	25
U2	10	5186	0	5	1	2	3	0	6	0	0
U3	0	0	2477	2	0	2	7	0	8	1	6
U4	20	2	2	5171	7	16	14	2	72	61	65
U5	1	0	2	6	4642	0	4	8	11	56	14
U6	2	0	5	33	1	3695	5	0	14	6	1
U7	7	2	15	43	5	5	5162	0	15	2	3
U8	0	2	0	6	0	0	0	4635	0	4	3
U9	2	7	2	77	3	10	11	2	7505	4	37
U10	13	0	5	51	32	0	4	1	10	5317	23
U11	20	7	9	77	5	0	12	5	31	49	6258

(a)



	U1	U2	U3	U4	U5	U6	U7	U8	U9	U10	U11
U1	8724	21	1	81	39	8	19	14	23	32	84
U2	36	5082	0	7	0	3	22	4	48	1	10
U3	1	0	2297	44	0	40	22	13	73	10	3
U4	106	26	73	4292	13	31	104	14	288	176	309
U5	45	1	1	9	4334	5	14	7	83	220	25
U6	18	4	39	55	25	3467	42	1	39	5	67
U7	7	17	15	82	6	33	4925	1	60	46	67
U8	21	0	1	10	0	3	0	4590	19	0	6
U9	36	35	37	187	65	42	57	28	6908	72	193
U10	54	9	21	209	166	6	85	2	28	4680	196
U11	128	18	21	228	10	60	53	18	190	204	5543

(b)

Figure 4.6: Confusion matrix of the two-stream multi-resolution fusion architecture. (a) UCF (64,32) and (b) UCF (32,16).

Figure 4.4 illustrates the model loss with respect to epoch while training the proposed fusion architecture on KTH and UCF11 datasets. The KTH (64,32) achieved the lowest loss at end of training. However, the training loss for KTH (32, 16) was similar to KTH (64,32) around 500^{th} epoch. As the KTH (64,32) and KTH (32,16) models both reached similar levels of loss while training, their overall performance metric on the test dataset was similar. For the UCF dataset, the UCF (64,32) model reached significantly lower levels of loss than the UCF (32,16) dataset. Thus, the UCF (64,32) model performed better than the UCF (32,16) model in the test dataset.

Figure 4.5 represents the confusion matrix for the best KTH (64,32) and KTH (32,16) models. The KTH (64,32) and KTH (32,16) models both misclassified 547 and 533 samples of the boxing activity. They misclassified instances of boxing as walking. KTH (64,32) model correctly identified more instances of handwaving than KTH (32,16) model. KTH (32,16) model correctly identified more instances of walking compared to KTH (64,32) model. KTH (64,32) model misclassified 113 instances of handwaving as walking, whereas the KTH (32,16) model incorrectly classified 81 instances.

Figure 4.6 represents the confusion matrix for the best UCF (64,32) and UCF (32,16) models. UCF (64,32) model misclassified 77 instances of golf_swing as trampoline_jumping, 77 instances of golf_swing as walking, and 71 instances of trampoline_jumping as golf_swing. UCF (32,16) model incorrectly classified 309 instances of walking as golf_swing, 288 instances of trampoline_jumping as golf_swing, and 228 instances of golf_swing and walking.

Table 4.5: Percentile performance metrics of the proposed multi-resolution architecture for each activity in KTH dataset

Activity	**Model**	**Performance (%)**			**PTQ(%)**			**QAT(%)**		
		P	***R***	***F1***	***P***	***R***	***F1***	***P***	***R***	***F1***
$K1$	KTH (64,32)	94.2	99.9	97.0	94.3	99.9	97.0	94.2	99.9	97.0
	KTH (32,16)	92.5	99.9	96.0	92.4	99.9	96.0	92.5	99.9	96.0
$K2$	KTH (64,32)	99.2	99.9	99.5	99.2	99.9	99.5	99.2	99.9	99.5
	KTH (32,16)	98.0	99.9	98.9	98.0	99.9	98.9	98.0	99.9	98.9
$K3$	KTH (64,32)	98.8	99.9	99.3	98.8	99.8	99.3	98.8	99.9	99.3
	KTH (32,16)	99.1	99.4	99.2	99.1	99.3	99.2	99.1	99.4	99.2
$K4$	KTH (64,32)	100	94.5	97.2	100	94.5	97.2	100	94.5	97.2
	KTH (32,16)	100	93.0	96.4	100	93.3	96.5	100	93.0	96.4

* P = Precision, R = Recall, F1 = F1-Score

Table 4.6: Percentile performance metrics of the proposed multi-resolution architecture for each activity in UCF dataset

Activity	**Model**	**Performance (%)**			**PTQ(%)**			**QAT(%)**		
		P	***R***	***F1***	***P***	***R***	***F1***	***P***	***R***	***F1***
$U1$	UCF (64,32)	99.1	98.4	98.7	99.1	98.3	98.7	99.0	99.1	99.0
	UCF (32,16)	95.0	96.4	95.7	95.0	96.3	95.6	96.6	96.7	96.7
$U2$	UCF (64,32)	99.5	99.4	99.5	99.5	99.4	99.4	99.3	99.7	99.5
	UCF (32,16)	97.4	97.4	97.4	97.4	97.4	97.3	98.7	98.4	98.5
$U3$	UCF (64,32)	98.2	98.9	98.6	98.2	98.8	98.4	98.5	99.0	98.8
	UCF (32,16)	91.6	91.7	91.7	91.6	91.7	91.6	94.9	95.4	95.2
$U4$	UCF (64,32)	93.4	95.1	94.2	93.5	95.1	94.2	94.3	97.0	95.6
	UCF (32,16)	82.4	79.0	80.7	82.4	79.0	80.7	88.7	84.3	86.4
$U5$	UCF (64,32)	98.6	97.8	98.2	98.6	97.9	98.2	98.9	98.6	98.7
	UCF (32,16)	93.0	91.3	92.1	93.0	91.7	92.3	89.9	95.6	92.6
$U6$	UCF (64,32)	98.9	98.2	98.5	99.0	98.2	98.5	99.1	98.6	98.9
	UCF (32,16)	93.7	92.1	92.9	93.7	92.5	93.1	97.7	91.6	94.5
$U7$	UCF (64,32)	98.7	98.1	98.4	98.7	98.1	98.4	98.8	98.3	98.5
	UCF (32,16)	92.1	93.6	92.9	92.1	93.6	92.9	93.4	95.0	94.2
$U8$	UCF (64,32)	99.5	99.6	99.5	99.5	99.6	99.5	99.4	99.7	99.6
	UCF (32,16)	97.8	98.7	98.2	97.9	98.6	98.2	98.8	99.0	98.9
$U9$	UCF (64,32)	97.7	97.9	97.8	97.7	97.9	97.7	97.2	98.9	98.0
	UCF (32,16)	89.0	90.1	89.6	89.0	90.1	89.5	94.2	89.8	91.9
$U10$	UCF (64,32)	96.3	97.4	96.9	96.3	97.4	96.9	98.0	97.3	97.6
	UCF (32,16)	85.9	85.7	85.8	85.9	85.7	85.8	81.5	93.7	87.2
$U11$	UCF (64,32)	97.2	96.6	96.9	97.2	96.6	96.9	98.9	95.2	97.1
	UCF (32,16)	85.2	85.6	85.4	85.2	85.6	85.4	90.9	86.0	88.4

* P = Precision, R = Recall, F1 = F1-Score

Table 4.5 presents the performance of the proposed fusion approach and its quantized versions on KTH dataset in terms of precision, recall, and F1-Score for each activity. The highest difference of precision was in $K1$ activity where the precision of KTH (32,16) model is lower than KTH (64,32) model. The activity $K4$ had the lowest recall of 97.2% and 96.4% for KTH (64,32) and KTH (32,16) models. However, $K4$ activity had the highest precision of 100% for both models. In general, the performance of PTQ and QAT models were similar for all activities in both models. The highest F1-Score of 99.2% was achieved for the $K2$ activity by the KTH

Table 4.7: Recognition performance of the proposed two-stream multi-resolution fusion approach

Model	**PM**	**A**	**P**	**R**	**F1**	**S**
KTH (64,32)	32,308	98.3	98.3	98.2	98.2	1700
KTH (32,16)	7,732	98.2	98.3	98.2	98.2	575.7
UCF (64,32)	32,308	97.9	97.9	97.9	97.9	1700
UCF (32,16)	7,732	91.1	91.0	91.1	91.0	576.5

PM = Parameters, A = Accuracy, P = Precision, R = Recall
F1 = F1-Score, S = Size in KB.

(64,32) model.

Table 4.6 presents the per-activity performance of the proposed fusion approach and its quantized versions on UCF dataset in terms of precision, recall, and F1-Score. The UCF (64,32) model obtained the highest precision of 99.5% in $U2$ activity, best recall of 99.6% in $U8$ activity, and highest F1-Score of 99.5% in $U2$ and $U8$ activities. The UCF (64,32) model achieved the lowest precision, recall, and F1-Score of 93.4%, 95.1% and 94.2% in $U4$ activity. The UCF (32,16) model obtained the highest precision of 97.8% in $U8$ activity, highest recall of 98.7% in $U8$ activity, and best F1-Score of 98.2% in $U8$ activity. The UCF (32,16) model achieved the lowest precision, recall, and F1-Score of 82.4%, 79.0%, and 80.7% in $U4$ activity. For UCF (32,16) model, in activities: $U1$, $U2$, $U3$, $U4$, $U10$, and $U11$ - the QAT models outperformed PTQ models in terms of precision, recall, and F1-Score. For UCF (64,32) model, the performance of the QAT models were similar to PTQ models across almost all activities.

Table 4.7 represents the recognition performance of the proposed two-stream multi-resolution fusion architecture on the KTH and UCF11 datasets. For the KTH dataset, the performance was similar for both KTH (64,32) and KTH (32,16) models. The accuracy for the KTH (64,32) model was marginally higher, while the original size of KTH (64,32) model was 2.9 times than the size of KTH (32,16) model. For the UCF11 dataset, the accuracy, precision, recall, and F1-Score were significantly higher for the UCF (64,32) model. The KTH (64,32) model achieved 97.9% performance score in all metrics, whereas the KTH (32,16) model achieved accuracy, precision, recall, and F1-Score of 91.1%, 91.0%, 91.1%, and 91.0%, respectively. The UCF models have a size relationship between them, similar to the KTH models.

Table 4.8 presents the performance of the developed fusion networks after quantization. As PTQ converts the already trained model, the performance metrics are very similar to the original model. QAT trains a new model while taking into account the quantization loss with the model loss. QAT models can have better or worse performance than the original model. The PTQ KTH

Table 4.8: Experimental results of the quantized multi-resolution fusion architecture

Model	**PM**	**PTQ(%)**					**QAT(%)**				
		A	***P***	***R***	***F1***	***S***	***A***	***P***	***R***	***F1***	***S***
KTH (64,32)	32,308	98.3	98.3	98.2	98.2	453.5	98.2	98.3	98.2	98.2	121.3
KTH (32,16)	7,732	98.2	98.3	98.2	98.2	101.6	97.9	98.1	98.0	98.0	33.4
UCI (64,32)	32,308	97.9	97.9	97.9	97.9	454.7	97.8	97.9	97.8	97.8	121.4
UCI (32,16)	7,732	91.1	91.1	91.1	91.1	101.8	91.4	91.4	91.4	91.3	33.5

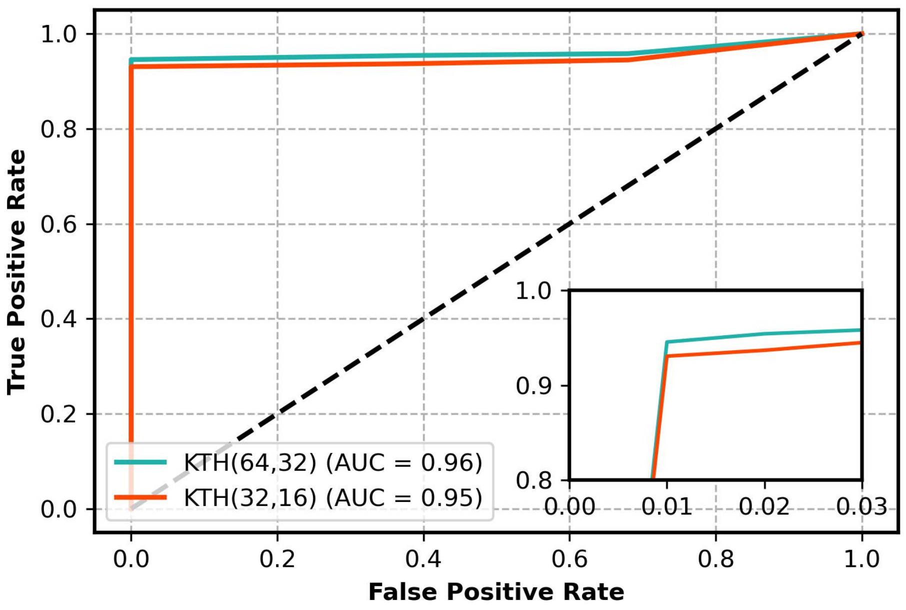
PM = Parameters, A = Accuracy, P = Precision, R = Recall, F1 = F1-Score, S = Size in KB.

Table 4.9: Delta of performance metrics between QAT and PTQ models

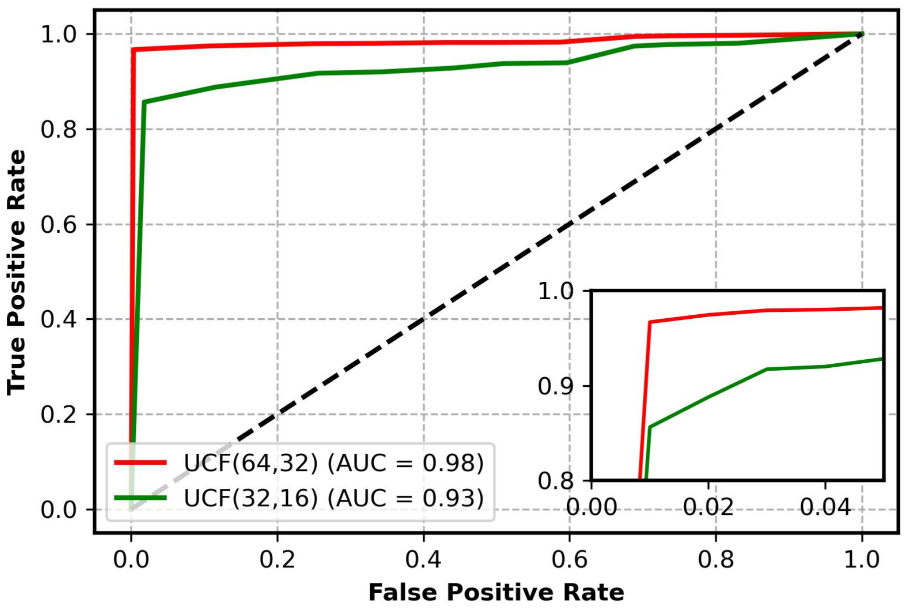
Model	**ΔA**	**ΔP**	**ΔR**	**ΔF1**	**ΔS**
KTH (64,32)	-0.09	0.00	0.00	0.00	-332.29
KTH (32,16)	-0.29	-0.20	-0.20	-0.20	-68.19
UCF (64,32)	-0.10	0.00	-0.10	-0.10	-333.29
UCF (32,16)	0.30	0.30	0.30	0.20	-68.39

(64,32) model had higher accuracy than the QAT KTH (64,32) model. However, the precision, recall, and F1-Score for both models are pretty similar. The QAT KTH (32,16) model has a 27.5% lower space requirement than the QAT KTH (64,32) model while using 23.9% lower parameters. The PTQ KTH (32,16) model uses 22.4% lower space than the PTQ KTH (32,16) model. For the UCF dataset, the QAT UCF (32,16) model performed marginally better than the PTQ UCF (32,16) model. Similar to the QAT UCF models, the QAT UCF (32,16) model has 27.5% lower space requirements than QAT UCF (64,32) model. However, in all cases of KTH and UCF models, QAT resulted in a significantly lower model size.

Table 4.9 represents the delta of performance metrics between the QAT and PTQ models. This helps us better to understand which quantization metric is providing better performance on average. We calculated the delta for each metric using (4.6). Here, X represents the performance metric, $\overline{X_{QAT}}$ represents the average value of the metric across 5 runs for the QAT model, $\overline{X_{PTQ}}$ represents the average values of the metric across 5 runs for the PTQ model. For accuracy, precision, recall, and F1-Score, a negative delta indicates that the PTQ models had a better overall performance. For size, a negative delta indicates that the PTQ models have higher memory requirements than the QAT models. For KTH (64,32), KTH (32,16), and UCF (64,32) models, the accuracy, precision, recall, and F1-Score are mostly negative. Only for UCF (32,16) model, the delta scores are positive. Thus, on average PTQ method lead to better-performing models.



(a)



(b)

Figure 4.7: Performance measures (ROC curves) of the developed two-stream mutti-resolution fusion approach. (a) KTH (b) UCF11.

However, all of the models had negative size delta, meaning the QAT method had led to better storage-optimized models while having slightly worse performance. Ultimately, the selection of a quantization method depends on a multiple factors: the task, the development or deployment board, the model, etc. If the development board has very little onboard memory, the QAT models are more suitable. Also, in some cases, QAT models might outperform PTQ models. Thus, it is always recommended to train the QAT models and compare their performance with PTQ models.

$$\Delta X = \overline{X_{QAT}} - \overline{X_{PTQ}} \tag{4.6}$$

Figure 4.7 presents the Receiver Operating Characteristic (ROC) curve of the proposed fusion architecture on KTH and UCF11 datasets. The solid lines with different color represent the Area Under Curve (AUC) values of each model across all activities. Generally a higher AUC score indicates a better performing model. As the KTH (64,32) and KTH (32,16) models perform similarly across all activities, their AUC scores are similar. The rectangular shape of both lines indicates that the models are generalizing and performing well. The UCF (64,32) model has a significantly higher AUC score of 0.98 than the UCF (32,16) model (AUC: 0.93) indicating the former model is vastly outperforming the latter. The almost rectangular shape of UCF (64,32) model also indicates a well-performing model.

4.5.2 Ablation Studies

Table 4.10 presents the experimental findings of different variants of the proposed multi-resolution fusion architecture. Variations include context-only stream, fovea-only stream, and two-stream reverse-resolution architecture.

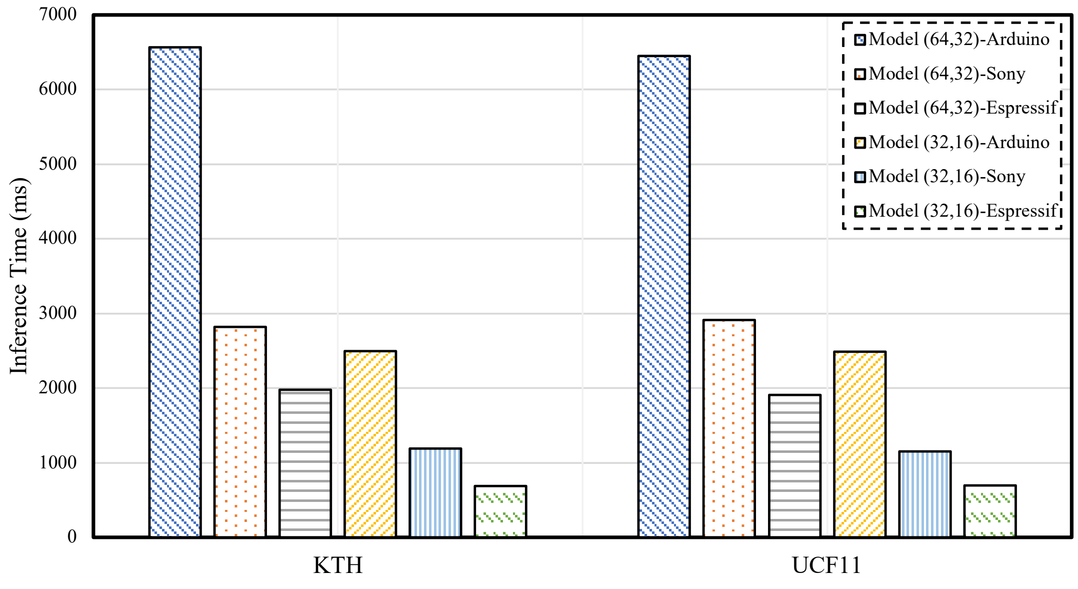
Context-only Stream

KTH (64,0), KTH (32,0), UCF (64,0), and UCF (32,0) represent the context-only stream architecture. In these models, only the full images were used as input. The fovea stream was not used. KTH (64,0) performed better than KTH (32,0) model. In both cases, the QAT variants performed marginally better than the PTQ variants. UCF (64,0) performed significantly better than UCF (32,0). The QAT variants of UCF model performed better comapred to the PTQ variants. Both the KTH and UCF models could not outperform the proposed two-stream multi-resolution fusion networks.

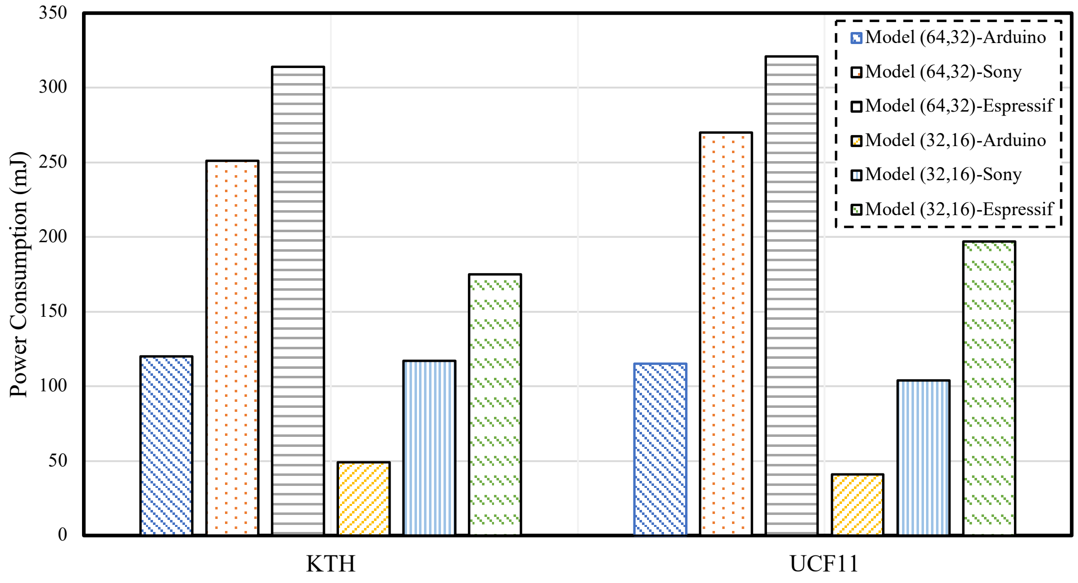
Table 4.10: Experimental findings of different variants of the proposed multi-resolution fusion architecture

Model	**PM**	**Performance (%)**					**PTQ(%)**					**QAT(%)**				
		A	***P***	***R***	***F1***	***S***	***A***	***P***	***R***	***F1***	***S***	***A***	***P***	***R***	***F1***	***S***
KTH (64,0)	26,452	96.0	96.3	96.0	95.9	1500	96.0	96.3	96.0	95.9	428.9	96.3	96.6	96.3	96.3	112
KTH (32,0)	5,972	95.7	96.0	95.7	95.7	459.3	95.7	96.1	95.7	95.7	93.0	95.9	96.2	95.9	95.9	27.9
KTH (0,64)	26,452	95.2	95.6	95.2	95.2	457.7	95.2	95.6	95.2	95.2	92.9	95.8	96.1	95.8	95.8	27.9
KTH (0,32)	5,972	89.9	90.9	89.9	89.9	238.1	89.9	90.9	89.9	89.9	20.9	90.7	91.5	90.7	90.7	9.9
KTH (32,64)	32,308	96.6	96.8	96.6	96.6	671.6	96.6	96.8	96.6	96.6	131.2	96.7	97.0	96.7	96.7	40.8
KTH (16,32)	7,732	93.8	94.3	93.9	93.8	376.6	93.9	94.3	93.9	93.8	35.2	94.0	94.5	94.0	94.0	16.8
UCF (64,0)	26,452	94.6	94.6	94.6	94.6	507.4	94.6	94.6	94.6	94.6	106.8	95.6	95.6	95.6	95.6	31.5
UCF (32,0)	5,972	82.1	82.1	82.2	82.1	263.3	82.2	82.1	82.2	82.0	26.9	83.1	83.0	83.1	82.8	11.5
UCF (0,64)	26,452	69.1	69.4	69.1	68.9	261.7	69.1	69.4	69.1	68.9	26.8	70.3	70.1	70.3	70.0	11.5
UCF (0,32)	5,972	49.4	50.0	49.4	48.4	214.1	49.4	50.0	49.4	48.4	10.9	50.7	51.0	50.7	49.9	7.59
UCF (32,64)	32,308	80.0	80.0	80.0	79.7	672.4	80.0	79.8	80.0	79.7	131.4	81.5	81.6	81.5	81.4	40.9
UCF (16,32)	7,732	58.6	58.4	58.6	57.3	214.1	58.6	58.5	58.6	57.3	35.4	61.4	60.5	61.4	60.5	16.9

* PM = Parameters, A = Accuracy, P = Precision, R = Recall, F1 = F1-Score, S = Size in KB.



(a)



(b)

Figure 4.8: Performance of the proposed two-stream multi-resolution fusion architecture on the tested boards. (a) inference time (b) power consumption.

Fovea-only Stream

KTH (0,64), KTH (0,32), UCF (0,64), and UCF (0,32) models represent the fovea-only stream architecture. The models only take the cropped center images as input. The context streams in these models are not used. KTH (0,64) performed significantly better than KTH (0,32). In both cases, the QAT variants of KTH performed marginally better than the PTQ variants. For UCF models, UCF (0,64) performed significantly better compared to UCF (0,32). However, both of them performed worse than other ablation models. The QAT variants of UCF models outperformed the PTQ variants.

Two-stream Reverse-resolution

KTH (32,64), KTH (16,32), UCF (32,64), and UCF (16,32) models represent the two-stream reverse-resolution architecture. In this architecture, the fovea stream receives the lower-resolution full images as input and the context stream receives the higher-resolution center-cropped images. KTH (32,64) achieved higher levels of performance than KTH (16,32). The QAT variants of KTH models achieved marginally higher performance score compared to PTQ variants. Similarly, UCF (32,64) achieved much better performance than UCF (16,32). The QAT UCF variants outperformed the PTQ UCF variants.

Within all KTH variants, the two-stream reverse-resolution variant KTH (32,64) achieved the highest levels of accuracy, precision, recall, and F1-Score of 96.6%, 96.8%, 96.6%, and 96.6%, respectively. Within all UCF variants, the context-only stream UCF (64,0) achieved the highest performance - 94.6% in accuracy, precision, recall, and F1-Score. In all of the variant models, the QAT variants outperformed their PTQ counterparts.

4.5.3 Inference Time and Power Consumption

Figure 4.8 presents the performance of the proposed fusion architecture in terms of their inference time and power consumption in the three tested boards. For any specific device, the inference time and power consumption of a model depend on the following: the number of convolutions, the size of fully connected layers, and the input size. As we used the same model with a fixed number of convolutions and fully connected layers, the input size mainly dictated the inference time and power consumption. The inference time for similar models for both KTH and UCF11 datasets is similar. For KTH models, the KTH (64,32) model variants have higher inference times than KTH (32,16) models. For KTH (64,32) model, the Arduino board had the highest inference time, followed by the Sony board, and the Espressif board had the lowest inference time. This is due to the different types of processors used in each board. The board with a higher clock speed processor can retrieve and interpret instructions quicker than a board with a lower clock speed, leading to better inference time performance. Similarly, UCF (64,32) had a higher average inference time than UCF (32,16) models. To the end, the Arduino board had the highest inference time, followed by the Sony board, with the Espressif board having the lowest inference time.

Although higher clock speed results in better inference time performance, they also require a higher voltage to run, leading to an overall increase in power consumption. The UCF (64,32) and KTH (64,32) models had higher power consumption than UCF (32,16) and KTH (32,16) models. Within the KTH models, the Arduino board had the lowest power consumption, followed by the Sony board, and the Espressif board had the highest power consumption. Similar trends can be noticed in the UCF models.

In all of the tested models and their variants, the higher-resolution models performed better in terms of accuracy, precision, recall, and F1-Score than their lower-resolution counterparts. This can be explained by the loss of information during downsampling. As we resized an image to a lower resolution, some useful shape information can be lost that could have been crucial for detection. However, higher-resolution models require higher inference and higher power requirements than lower-resolution models. Through our experiments, it is visible that, instead of using a single-stream QAT KTH (64,0) model, our proposed two-stream QAT KTH (32,16) model results in 29.2% parameter reduction and 29.8% size reduction while increasing accuracy, precision, recall, and F1-Score by 2.3%, 2.0%, 2.2%, and 2.3%, respectively.

4.6 Summary

In this chapter, we introduced the multimodal datasets used in our research. We presented the two types of performance metrics that we used to evaluate our models: model performance in terms of the number of parameters, size, accuracy, precision, recall, and F1-Score and device performance in terms of inference time and power consumption. We reported and analyzed the benchmarking performance of the CNN and DSCNN models in the multimodal datasets. We also presented the performance and ablation studies of our proposed two-stream multi-resolution fusion architecture. The quantitative and qualitative analysis of the research presented in this chapter indicates the feasibility of efficient HAR systems in edge devices.

Chapter 5

Conclusion and Future Works

This chapter presents the concluding remarks, limitations, and future research scopes of our presented research work in this thesis.

5.1 Conclusion

TinyML is a rapidly evolving research domain with numerous applications that can change our lives for the better. In this thesis, at first, we presented a benchmark study to evaluate the performance of standard CNN and DSCNN models using five popular human activity recognition benchmark datasets with various data modalities on low-power devices. We also evaluated the performance of QAT and PTQ training methodologies for quantization. We evaluated model performance based on their number of parameters, accuracy, precision, recall, F1-Score, and size in KB. We deployed the model-based applications to multiple publicly available resource-constrained edge devices and evaluated the applications in terms of their inference time and power consumption. The outcomes of each model are highly diverse depending on the model size, quantization, input size, and structure. Later, we proposed a two-stream multi-resolution fusion architecture for human activity recognition from video data modality in resource-constrained devices. We tested the proposed model in two challenging datasets: KTH and UCF. We evaluated the performance of two quantization methods: QAT and PTQ on the proposed architecture. We deployed the proposed QAT models on commodity edge hardware and evaluated the performance in terms of inference time and power consumption.

It is evident from the experimental results that current methods available through advances in TinyML are suitable for human activity recognition on low-power devices and are ready to use for real-time application purposes.

5.2 Limitations and Future Works

While TinyML is rapidly evolving, there are several limitations the research community needs to focus on to make TinyML ubiquitous. Developing and deploying TinyML models on edge devices require specific low-level programming for each different type of device. This lack of interoperability and accessibility are significant roadblocks to the mass adaptation of TinyML. Although the proposed fusion architecture performed very well in detecting human activities, it still could not take advantage of the temporal features of video streams. A combination of the proposed model with recurrent models such as RNN or LSTM could improve the performance. However, currently, the publicly available TinyML libraries do not support recurrent networks for a wide number of commodity devices.

In the future, more diverse generalized neural architectures such as LSTM and RNN can be developed and tested in System on Chip (SoC) and other resource constrained devices. Efforts can be focused on creating indoor activity-oriented datasets by collecting and augmenting data from multiple datasets. Creating an open-access general model trained on multiple different HAR video datasets would provide out-of-the-box framework support for smart Tiny HAR applications. Creating no-code or low-code TinyML deployment solutions for a wide range of edge devices would further reduce the barrier to entry and improve accessibility.

References

[1] Sen Qiu, Hongkai Zhao, Nan Jiang, Zhelong Wang, Long Liu, Yi An, Hongyu Zhao, Xin Miao, Ruichen Liu, and Giancarlo Fortino. Multi-sensor information fusion based on machine learning for real applications in human activity recognition: State-of-the-art and research challenges. *Information Fusion*, 80:241–265, 2022.

[2] Md Milon Islam, Sheikh Nooruddin, Fakhri Karray, and Ghulam Muhammad. Human activity recognition using tools of convolutional neural networks: A state of the art review, data sets, challenges, and future prospects. *Computers in Biology and Medicine*, page 106060, 2022.

[3] Zehua Sun, Qiuhong Ke, Hossein Rahmani, Mohammed Bennamoun, Gang Wang, and Jun Liu. Human action recognition from various data modalities: A review. *IEEE transactions on pattern analysis and machine intelligence*, 2022.

[4] Kaixuan Chen, Dalin Zhang, Lina Yao, Bin Guo, Zhiwen Yu, and Yunhao Liu. Deep learning for sensor-based human activity recognition: Overview, challenges, and opportunities. *ACM Computing Surveys (CSUR)*, 54(4):1–40, 2021.

[5] Preksha Pareek and Ankit Thakkar. A survey on video-based human action recognition: recent updates, datasets, challenges, and applications. *Artificial Intelligence Review*, 54(3):2259–2322, 2021.

[6] Iqbal H. Sarker. Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions. *Sn Computer Science*, 2(6):420, 2021.

[7] Iqbal H. Sarker. Machine learning: Algorithms, real-world applications and research directions. *Sn Computer Science*, 2(3):160, 2021.

[8] Shibo Zhang, Yaxuan Li, Shen Zhang, Farzad Shahabi, Stephen Xia, Yu Deng, and Nabil Alshurafa. Deep learning in human activity recognition with wearable sensors: A review on advances. *Sensors*, 22(4):1476, 2022.

[9] Farzana Kulsoom, Sanam Narejo, Zahid Mehmood, Hassan Nazeer Chaudhry, Ayesha Butt, and Ali Kashif Bashir. A review of machine learning-based human activity recognition for diverse applications. *Neural Computing and Applications*, pages 1–36, 2022.

[10] Ariza-Colpas P Patricia, Vicario Enrico, Butt Aziz Shariq, De-la_Hoz-Franco Emiro, Piñeres-Melo Marlon Alberto, Oviedo-Carrascal Ana Isabel, Muhammad Imran Tariq, Johanna Karina García Restrepo, and Patara Fulvio. Machine learning applied to datasets of human activity recognition: Data analysis in health care. *Current Medical Imaging*, 19(1):46–64, 2023.

[11] Yang Li, Guanci Yang, Zhidong Su, Shaobo Li, and Yang Wang. Human activity recognition based on multienvironment sensor data. *Information Fusion*, 91:47–63, 2023.

[12] Neha Gupta, Suneet K Gupta, Rajesh K Pathak, Vanita Jain, Parisa Rashidi, and Jasjit S Suri. Human activity recognition in artificial intelligence framework: A narrative review. *Artificial intelligence review*, 55(6):4755–4808, 2022.

[13] Md Milon Islam, Sheikh Nooruddin, Fakhri Karray, and Ghulam Muhammad. Multi-level feature fusion for multimodal human activity recognition in internet of healthcare things. *Information Fusion*, 94:17–31, 2023.

[14] Fuqiang Gu, Mu-Huan Chung, Mark Chignell, Shahrokh Valaee, Baoding Zhou, and Xue Liu. A survey on deep learning for human activity recognition. *ACM Computing Surveys (CSUR)*, 54(8):1–34, 2021.

[15] E Ramanujam, Thinagaran Perumal, and S Padmavathi. Human activity recognition with smartphone and wearable sensors using deep learning techniques: A review. *IEEE Sensors Journal*, 21(12):13029–13040, 2021.

[16] Muhammad Shafique, Theocharis Theocharides, Vijay Janapa Reddy, and Boris Murmann. Tinyml: Current progress, research challenges, and future roadmap. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 1303–1306. IEEE, 2021.

[17] Lachit Dutta and Swapna Bharali. Tinyml meets iot: A comprehensive survey. *Internet of Things*, 16:100461, 2021.

[18] Partha Pratim Ray. A review on tinyml: State-of-the-art and prospects. *Journal of King Saud University-Computer and Information Sciences*, 2021.

[19] Vijeta Sharma, Manjari Gupta, Anil Kumar Pandey, Deepti Mishra, and Ajai Kumar. A review of deep learning-based human activity recognition on benchmark video datasets. *Applied Artificial Intelligence*, 36(1):2093705, 2022.

[20] Ivan Miguel Pires, Faisal Hussain, Gonçalo Marques, and Nuno M Garcia. Comparison of machine learning techniques for the identification of human activities from inertial sensors available in a mobile device after the application of data imputation techniques. *Computers in Biology and Medicine*, 135:104638, 2021.

[21] Arti Maurya, Ram Kumar Yadav, and Manoj Kumar. Comparative study of human activity recognition on sensory data using machine learning and deep learning. In *Proceedings of Integrated Intelligence Enable Networks and Computing: IIENC 2020*, pages 63–71. Springer, 2021.

[22] Fatima Alshehri and Ghulam Muhammad. A comprehensive survey of the internet of things (iot) and ai-based smart healthcare. *IEEE Access*, 9:3660–3678, 2020.

[23] Darshan Vishwasrao Medhane, Arun Kumar Sangaiah, M Shamim Hossain, Ghulam Muhammad, and Jin Wang. Blockchain-enabled distributed security framework for next-generation iot: An edge cloud and software-defined network-integrated approach. *IEEE Internet of Things Journal*, 7(7):6143–6149, 2020.

[24] Abdu Gumaei, Mohammad Mehedi Hassan, Abdulhameed Alelaiwi, and Hussain Alsalman. A hybrid deep learning model for human activity recognition using multimodal body sensing data. *IEEE Access*, 7:99152–99160, 2019.

[25] Tanvir Mahmud, AQM Sazzad Sayyed, Shaikh Anowarul Fattah, and Sun-Yuan Kung. A novel multi-stage training approach for human activity recognition from multimodal wearable sensor data using deep neural network. *IEEE Sensors Journal*, 21(2):1715–1726, 2020.

[26] Reem Abdel-Salam, Rana Mostafa, and Mayada Hadhood. Human activity recognition using wearable sensors: review, challenges, evaluation benchmark. In *International Workshop on Deep Learning for Human Activity Recognition*, pages 1–15. Springer, 2021.

[27] Fatemeh Serpush, Mohammad Bagher Menhaj, Behrooz Masoumi, and Babak Karasfi. Wearable sensor-based human activity recognition in the smart healthcare system. *Computational Intelligence and Neuroscience*, 2022, 2022.

[28] Amira Mimouna and Anouar Ben Khalifa. A survey of human action recognition using accelerometer data. *Advanced Sensors for Biomedical Applications*, pages 1–32, 2021.

[29] Praveen Kumar Shukla, Ankit Vijayvargiya, Rajesh Kumar, et al. Human activity recognition using accelerometer and gyroscope data from smartphones. In *2020 International*

Conference on Emerging Trends in Communication, Control and Computing (ICONC3), pages 1–6. IEEE, 2020.

[30] Abdul Kadar Muhammad Masum, Erfanul Hoque Bahadur, Ahmed Shan-A-Alahi, Md Akib Uz Zaman Chowdhury, Mir Reaz Uddin, and Abdullah Al Noman. Human activity recognition using accelerometer, gyroscope and magnetometer sensors: Deep neural network approaches. In *2019 10Th international conference on computing, communication and networking technologies (ICCCNT)*, pages 1–6. IEEE, 2019.

[31] Mahsa Sadat Afzali Arani, Diego Elias Costa, and Emad Shihab. Human activity recognition: A comparative study to assess the contribution level of accelerometer, ecg, and ppg signals. *Sensors*, 21(21):6997, 2021.

[32] Ku Nurhanim, I Elamvazuthi, LI Izhar, Genci Capi, and Steven Su. Emg signals classification on human activity recognition using machine learning algorithm. In *2021 8th NAFOSTED Conference on Information and Computer Science (NICS)*, pages 369–373. IEEE, 2021.

[33] Sojeong Ha, Jeong-Min Yun, and Seungjin Choi. Multi-modal convolutional neural networks for activity recognition. In *2015 IEEE International conference on systems, man, and cybernetics*, pages 3017–3022. IEEE, 2015.

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

[35] Jun-Ho Choi and Jong-Seok Lee. Embracenet: A robust deep learning architecture for multimodal classification. *Information Fusion*, 51:259–270, 2019.

[36] Jun-Ho Choi and Jong-Seok Lee. Confidence-based deep multimodal fusion for activity recognition. In *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*, pages 1548–1556, 2018.

[37] Chih-Ta Yen, Jia-Xian Liao, and Yi-Kai Huang. Human daily activity recognition performed using wearable inertial sensors combined with deep learning algorithms. *Ieee Access*, 8:174105–174114, 2020.

[38] Isah A Lawal and Sophia Bano. Deep human activity recognition with localisation of wearable sensors. *IEEE Access*, 8:155060–155070, 2020.

[39] Sojeong Ha and Seungjin Choi. Convolutional neural networks for human activity recognition using multiple accelerometer and gyroscope sensors. In *2016 international joint conference on neural networks (IJCNN)*, pages 381–388. IEEE, 2016.

[40] Zhaoyang Niu, Guoqiang Zhong, and Hui Yu. A review on the attention mechanism of deep learning. *Neurocomputing*, 452:48–62, 2021.

[41] Rebeen Ali Hamad, Masashi Kimura, Longzhi Yang, Wai Lok Woo, and Bo Wei. Dilated causal convolution with multi-head self attention for sensor human activity recognition. *Neural Computing and Applications*, 33(20):13705–13722, 2021.

[42] Kun Wang, Jun He, and Lei Zhang. Attention-based convolutional neural network for weakly labeled human activities' recognition with wearable sensors. *IEEE Sensors Journal*, 19(17):7598–7604, 2019.

[43] Tan-Hsu Tan, Ching-Jung Huang, Munkhjargal Gochoo, and Yung-Fu Chen. Activity recognition based on fr-cnn and attention-based lstm network. In *2021 30th Wireless and Optical Communications Conference (WOCC)*, pages 146–149. IEEE, 2021.

[44] Yanan Liu, Hao Zhang, Dan Xu, and Kangjian He. Graph transformer network with temporal kernel attention for skeleton-based action recognition. *Knowledge-Based Systems*, 240:108146, 2022.

[45] Hechuang Wang. Deeply-learned and spatial–temporal feature engineering for human action understanding. *Future Generation Computer Systems*, 123:257–262, 2021.

[46] Yair A Andrade-Ambriz, Sergio Ledesma, Mario-Alberto Ibarra-Manzano, Marvella I Oros-Flores, and Dora-Luz Almanza-Ojeda. Human activity recognition using temporal convolutional neural network architecture. *Expert Systems with Applications*, 191:116287, 2022.

[47] Suguo Zhu, Zhenying Fang, Yi Wang, Jun Yu, and Junping Du. Multimodal activity recognition with local block cnn and attention-based spatial weighted cnn. *Journal of Visual Communication and Image Representation*, 60:38–43, 2019.

[48] 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.

[49] Yin Tang, Lei Zhang, Qi Teng, Fuhong Min, and Aiguo Song. Triple cross-domain attention on human activity recognition using wearable sensors. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2022.

[50] Dipanwita Thakur and Suparna Biswas. Smartphone based human activity monitoring and recognition using ml and dl: a comprehensive survey. *Journal of Ambient Intelligence and Humanized Computing*, 11(11):5433–5444, 2020.

[51] Guan Yuan, Zhaohui Wang, Fanrong Meng, Qiuyan Yan, and Shixiong Xia. An overview of human activity recognition based on smartphone. *Sensor Review*, 2018.

[52] Marcin Straczkiewicz, Peter James, and Jukka-Pekka Onnela. A systematic review of smartphone-based human activity recognition methods for health research. *NPJ Digital Medicine*, 4(1):1–15, 2021.

[53] Bandar Almaslukh, Abdel Monim Artoli, and Jalal Al-Muhtadi. A robust deep learning approach for position-independent smartphone-based human activity recognition. *Sensors*, 18(11):3726, 2018.

[54] Song-Mi Lee, Sang Min Yoon, and Heeryon Cho. Human activity recognition from accelerometer data using convolutional neural network. In *2017 ieee international conference on big data and smart computing (bigcomp)*, pages 131–134. IEEE, 2017.

[55] Daniele Ravi, Charence Wong, Benny Lo, and Guang-Zhong Yang. A deep learning approach to on-node sensor data analytics for mobile or wearable devices. *IEEE journal of biomedical and health informatics*, 21(1):56–64, 2016.

[56] Daniele Ravi, Charence Wong, Benny Lo, and Guang-Zhong Yang. Deep learning for human activity recognition: A resource efficient implementation on low-power devices. In *2016 IEEE 13th international conference on wearable and implantable body sensor networks (BSN)*, pages 71–76. IEEE, 2016.

[57] Zanobya N Khan and Jamil Ahmad. Attention induced multi-head convolutional neural network for human activity recognition. *Applied Soft Computing*, 110:107671, 2021.

[58] Haoxi Zhang, Zhiwen Xiao, Juan Wang, Fei Li, and Edward Szczerbicki. A novel iot-perceptive human activity recognition (har) approach using multihead convolutional attention. *IEEE Internet of Things Journal*, 7(2):1072–1080, 2019.

[59] Ge Zheng. A novel attention-based convolution neural network for human activity recognition. *IEEE Sensors Journal*, 21(23):27015–27025, 2021.

[60] Ohoud Nafea, Wadood Abdul, Ghulam Muhammad, and Mansour Alsulaiman. Sensor-based human activity recognition with spatio-temporal deep learning. *Sensors*, 21(6):2141, 2021.

[61] Nitin Nair, Chinchu Thomas, and Dinesh Babu Jayagopi. Human activity recognition using temporal convolutional network. In *Proceedings of the 5th international Workshop on Sensor-based Activity Recognition and Interaction*, pages 1–8, 2018.

[62] Xinyu Li, Yuan He, and Xiaojun Jing. A survey of deep learning-based human activity recognition in radar. *Remote Sensing*, 11(9):1068, 2019.

[63] Ali Hanif, Muhammad Muaz, Azhar Hasan, and Muhammad Adeel. Micro-doppler based target recognition with radars: A review. *IEEE Sensors Journal*, 2022.

[64] Wenbin Ye, Haiquan Chen, and Bing Li. Using an end-to-end convolutional network on radar signal for human activity classification. *IEEE Sensors Journal*, 19(24):12244–12252, 2019.

[65] Wenbin Ye and Haiquan Chen. Human activity classification based on micro-doppler signatures by multiscale and multitask fourier convolutional neural network. *IEEE Sensors Journal*, 20(10):5473–5479, 2020.

[66] Haiquan Chen and Wenbin Ye. Classification of human activity based on radar signal using 1-d convolutional neural network. *IEEE Geoscience and Remote Sensing Letters*, 17(7):1178–1182, 2019.

[67] Ibrahim Alnujaim, Daegun Oh, and Youngwook Kim. Generative adversarial networks for classification of micro-doppler signatures of human activity. *IEEE Geoscience and Remote Sensing Letters*, 17(3):396–400, 2019.

[68] Baris Erol, Sevgi Z Gurbuz, and Moeness G Amin. Gan-based synthetic radar micro-doppler augmentations for improved human activity recognition. In *2019 IEEE Radar Conference (RadarConf)*, pages 1–5. IEEE, 2019.

[69] Chaoyang Wu and Wenbin Ye. Generative adversarial network for radar-based human activities classification with low training data support. In *2021 IEEE 4th International Conference on Electronic Information and Communication Technology (ICEICT)*, pages 415–419. IEEE, 2021.

[70] Ibrahim Alnujaim, Shobha Sundar Ram, Daegun Oh, and Youngwook Kim. Synthesis of micro-doppler signatures of human activities from different aspect angles using generative adversarial networks. *IEEE Access*, 9:46422–46429, 2021.

[71] Li-Fang Wu, Qi Wang, Meng Jian, Yu Qiao, and Bo-Xuan Zhao. A comprehensive review of group activity recognition in videos. *International Journal of Automation and Computing*, 18(3):334–350, 2021.

[72] Tej Singh and Dinesh Kumar Vishwakarma. Human activity recognition in video benchmarks: A survey. *Advances in Signal Processing and Communication*, pages 247–259, 2019.

[73] Malik Ali Gul, Muhammad Haroon Yousaf, Shah Nawaz, Zaka Ur Rehman, and Hyung-Won Kim. Patient monitoring by abnormal human activity recognition based on cnn architecture. *Electronics*, 9(12):1993, 2020.

[74] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[75] Shubham Shinde, Ashwin Kothari, and Vikram Gupta. Yolo based human action recognition and localization. *Procedia computer science*, 133:831–838, 2018.

[76] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. *Advances in neural information processing systems*, 27, 2014.

[77] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[78] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016.

[79] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2012.

[80] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Dense trajectories and motion boundary descriptors for action recognition. *International journal of computer vision*, 103(1):60–79, 2013.

[81] Limin Wang, Yu Qiao, and Xiaoou Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4305–4314, 2015.

[82] Fatemeh Serpush and Mahdi Rezaei. Complex human action recognition using a hierarchical feature reduction and deep learning-based method. *SN Computer Science*, 2(2):1–15, 2021.

[83] Jagadeesh Basavaiah and Chandrashekar Mohan Patil. Human activity detection and action recognition in videos using convolutional neural networks. *Journal of Information and Communication Technology*, 19(2):157–183, 2020.

[84] Vijay Janapa Reddi, Christine Cheng, David Kanter, Peter Mattson, Guenther Schmuelling, Carole-Jean Wu, Brian Anderson, Maximilien Breughe, Mark Charlebois, William Chou, et al. Mlperf inference benchmark. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pages 446–459. IEEE, 2020.

[85] Colby Banbury, Vijay Janapa Reddi, Peter Torelli, Nat Jeffries, Csaba Kiraly, Jeremy Holleman, Pietro Montino, David Kanter, Pete Warden, Danilo Pau, et al. Mlperf tiny benchmark. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, pages 1–15, 2021.

[86] Bharath Sudharsan, Simone Salerno, Duc-Duy Nguyen, Muhammad Yahya, Abdul Wahid, Piyush Yadav, John G Breslin, and Muhammad Intizar Ali. Tinyml benchmark: Executing fully connected neural networks on commodity microcontrollers. In *2021 IEEE 7th World Forum on Internet of Things (WF-IoT)*, pages 883–884. IEEE, 2021.

[87] Christos Profentzas, Magnus Almgren, and Olaf Landsiedel. Performance of deep neural networks on low-power iot devices. In *Proceedings of the workshop on benchmarking cyber-physical systems and Internet of things*, pages 32–37, 2021.

[88] Danilo Pau, Marco Lattuada, Francesco Loro, Antonio De Vita, and Gian Domenico Licciardo. Comparing industry frameworks with deeply quantized neural networks on microcontrollers. In *2021 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–6. IEEE, 2021.

[89] Swapnil Sayan Saha, Sandeep Singh Sandha, and Mani Srivastava. Machine learning for microcontroller-class hardware-a review. *IEEE Sensors Journal*, 2022.

[90] Sorin Zoican, Marius Vochin, Roxana Zoican, and Dan Galatchi. Neural network testing framework for microcontrollers. In *2022 14th International Conference on Communications (COMM)*, pages 1–6. IEEE, 2022.

[91] Shubham Gupta, Sweta Jain, Bholanath Roy, and Abhishek Deb. A tinyml approach to human activity recognition. In *Journal of Physics: Conference Series*, volume 2273, page 012025. IOP Publishing, 2022.

[92] Yexu Zhou, Haibin Zhao, Yiran Huang, Till Riedel, Michael Hefenbrock, and Michael Beigl. Tinyhar: A lightweight deep learning model designed for human activity recognition. In *Proceedings of the 2022 ACM International Symposium on Wearable Computers*, pages 89–93, 2022.

[93] Preeti Agarwal and Mansaf Alam. A lightweight deep learning model for human activity recognition on edge devices. *Procedia Computer Science*, 167:2364–2373, 2020.

[94] Hojjat Salehinejad and Shahrokh Valaee. Litehar: Lightweight human activity recognition from wifi signals with random convolution kernels. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4068–4072. IEEE, 2022.

[95] YL Coelho, B Nguyen, FA Santos, S Krishnan, and TF Bastos-Filho. A lightweight model for human activity recognition based on two-level classifier and compact cnn model. In *XXVII Brazilian Congress on Biomedical Engineering: Proceedings of CBEB 2020, October 26–30, 2020, Vitória, Brazil*, pages 1895–1901. Springer, 2022.

[96] Atis Elsts and Ryan McConville. Are microcontrollers ready for deep learning-based human activity recognition? *Electronics*, 10(21):2640, 2021.

[97] Tianyi Liu, Shuoyuan Wang, Yue Liu, Weiming Quan, and Lei Zhang. A lightweight neural network framework using linear grouped convolution for human activity recognition on mobile devices. *The Journal of Supercomputing*, pages 1–21, 2022.

[98] Wenbo Huang, Lei Zhang, Hao Wu, Fuhong Min, and Aiguo Song. Channel-equalization-har: a light-weight convolutional neural network for wearable sensor based human activity recognition. *IEEE Transactions on Mobile Computing*, 2022.

[99] Emanuele Lattanzi, Matteo Donati, and Valerio Freschi. Exploring artificial neural networks efficiency in tiny wearable devices for human activity recognition. *Sensors*, 22(7):2637, 2022.

[100] Ankita, Shalli Rani, Himanshi Babbar, Sonya Coleman, Aman Singh, and Hani Moaiteq Aljahdali. An efficient and lightweight deep learning model for human activity recognition using smartphones. *Sensors*, 21(11):3845, 2021.

[101] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern recognition*, 77:354–377, 2018.

[102] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[103] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[104] Andrinandrasana David Rasamoelina, Fouzia Adjailia, and Peter Sinčák. A review of activation function for artificial neural network. In *2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pages 281–286. IEEE, 2020.

[105] Manli Sun, Zhanjie Song, Xiaoheng Jiang, Jing Pan, and Yanwei Pang. Learning pooling for convolutional neural network. *Neurocomputing*, 224:96–104, 2017.

[106] SH Shabbeer Basha, Shiv Ram Dubey, Viswanath Pulabaigari, and Snehasis Mukherjee. Impact of fully connected layers on performance of convolutional neural networks for image classification. *Neurocomputing*, 378:112–119, 2020.

[107] Yury Nahshan, Brian Chmiel, Chaim Baskin, Evgenii Zheltonozhskii, Ron Banner, Alex M Bronstein, and Avi Mendelson. Loss aware post-training quantization. *Machine Learning*, 110(11-12):3245–3262, 2021.

[108] Ron Banner, Yury Nahshan, and Daniel Soudry. Post training 4-bit quantization of convolutional networks for rapid-deployment. *Advances in Neural Information Processing Systems*, 32, 2019.

[109] Eunhyeok Park, Sungjoo Yoo, and Peter Vajda. Value-aware quantization for training and inference of neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 580–595, 2018.

[110] Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart Van Baalen, and Tijmen Blankevoort. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*, 2021.

[111] Agus Kurniawan and Agus Kurniawan. Arduino nano 33 ble sense board development. *IoT Projects with Arduino Nano 33 BLE Sense: Step-By-Step Projects for Beginners*, pages 21–74, 2021.

[112] Marco Giordano, Luigi Piccinelli, and Michele Magno. Survey and comparison of milliwatts micro controllers for tiny machine learning at the edge. In *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pages 94–97. IEEE, 2022.

[113] Md Ziaul Haque Zim. Tinyml: analysis of xtensa lx6 microprocessor for neural network applications by esp32 soc. *arXiv preprint arXiv:2106.10652*, 2021.

[114] Lourdes Martínez-Villaseñor, Hiram Ponce, Jorge Brieva, Ernesto Moya-Albor, José Núñez-Martínez, and Carlos Peñafort-Asturiano. Up-fall detection dataset: A multimodal approach. *Sensors*, 19(9):1988, 2019.

[115] Kripesh Adhikari, Hamid Bouchachia, and Hammadi Nait-Charif. Activity recognition for indoor fall detection using convolutional neural network. In *2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA)*, pages 81–84. IEEE, 2017.

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

[117] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge L Reyes-Ortiz. Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In *International workshop on ambient assisted living*, pages 216–223. Springer, 2012.

[118] Jennifer R Kwapisz, Gary M Weiss, and Samuel A Moore. Activity recognition using cell phone accelerometers. *ACM SigKDD Explorations Newsletter*, 12(2):74–82, 2011.

[119] Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: a local svm approach. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 3, pages 32–36. IEEE, 2004.

[120] Jingen Liu, Jiebo Luo, and Mubarak Shah. Recognizing realistic actions from videos “in the wild”. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1996–2003. IEEE, 2009.