IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 9 June 2023, accepted 9 July 2023, date of publication 12 July 2023, date of current version 19 July 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3294564

RESEARCH ARTICLE

Efficient Human Activity Recognition Using Lookup Table-Based Neural Architecture Search for Mobile Devices

WON-SEON LIM[1], WANGDUK SEO[1], DAE-WON KIM[1], (Member, IEEE), AND JAESUNG LEE[2]

[1]School of Computer Science and Engineering, Chung-Ang University, Dongjak-gu, Seoul 06974, Republic of Korea
[2]Department of Artificial Intelligence, Chung-Ang University, Dongjak-gu, Seoul 06974, Republic of Korea

Corresponding authors: Dae-Won Kim (dwkim@cau.ac.kr) and Jaesung Lee (curseor@cau.ac.kr)

This work was supported in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant funded by the Korean Government (MSIT) (Development of Integrated Development Framework That Supports Automatic Neural Network Generation and Deployment Optimized for Runtime Environment) under Grant 2021-0-00766; in part by the Chung-Ang University Research Grants, in 2021; and in part by the National Research Foundation of Korea (NRF) Grant funded by the Korea Government (MSIT) under Grant 2023R1A2C1006745.

ABSTRACT Mobile devices play a crucial role in human activity recognition as they enable real-time sensing of user interaction for learning algorithms like neural networks. To facilitate human activity recognition on mobile devices, it is important to deploy efficient neural network architectures due to the limited computational capacity of these devices. However, conventional neural architecture search methods often generate less effective architectures because they neglect the specific requirements of target devices on which the neural network would operate in real-time. Moreover, these methods are impractical in the mobile device environment due to their high computational cost for architecture search. To address these challenges, we propose an efficient neural architecture search method based on a latency lookup table. Our proposed method efficiently performs the network search process based on differentiable NAS while considering the actual latency of mobile devices, which is stored in a lookup table. The experimental results on public datasets provide evidence that the proposed method outperforms conventional methods in terms of speed. We achieved a search time of under 1.5 hours on each dataset, which is more than seven times faster on average compared to conventional methods. Furthermore, our in-depth analysis shows that the optimal architecture can vary depending on the target mobile devices, such as Galaxy A31 and S10. By tailoring the models to each device, optimized models achieved an additional 4-5% improvement in inference time for each respective device.

INDEX TERMS Human activity recognition, deep learning, neural architecture search.

I. INTRODUCTION

Human activity recognition (HAR) using wearable and mobile devices has attracted considerable research attention for application in fields such as healthcare [1], surveillance [2], smart Home [3]. Many studies have focused on offline mobile device-based activity monitoring to address challenges such as privacy, communication cost, latency, and network traffic to the cloud [4]. However, mobile devices have limited resources and diverse hardware specifications, making the design of HAR-specific models crucial for effective activity recognition on these devices [5].

Deep learning algorithms have shown excellent performance in most HAR studies, particularly the DeepConvLSTM approach [6], which has achieved state-of-the-art results. This approach combines the strengths of Convolutional Neural Networks (CNNs) [7] and Recurrent Neural Networks (RNNs) [8], creating a hybrid network architecture. In particular, designing the CNN architecture effectively to extract valuable features plays a crucial role in performance [9].

The associate editor coordinating the review of this manuscript and approving it for publication was Chan Hwang See.

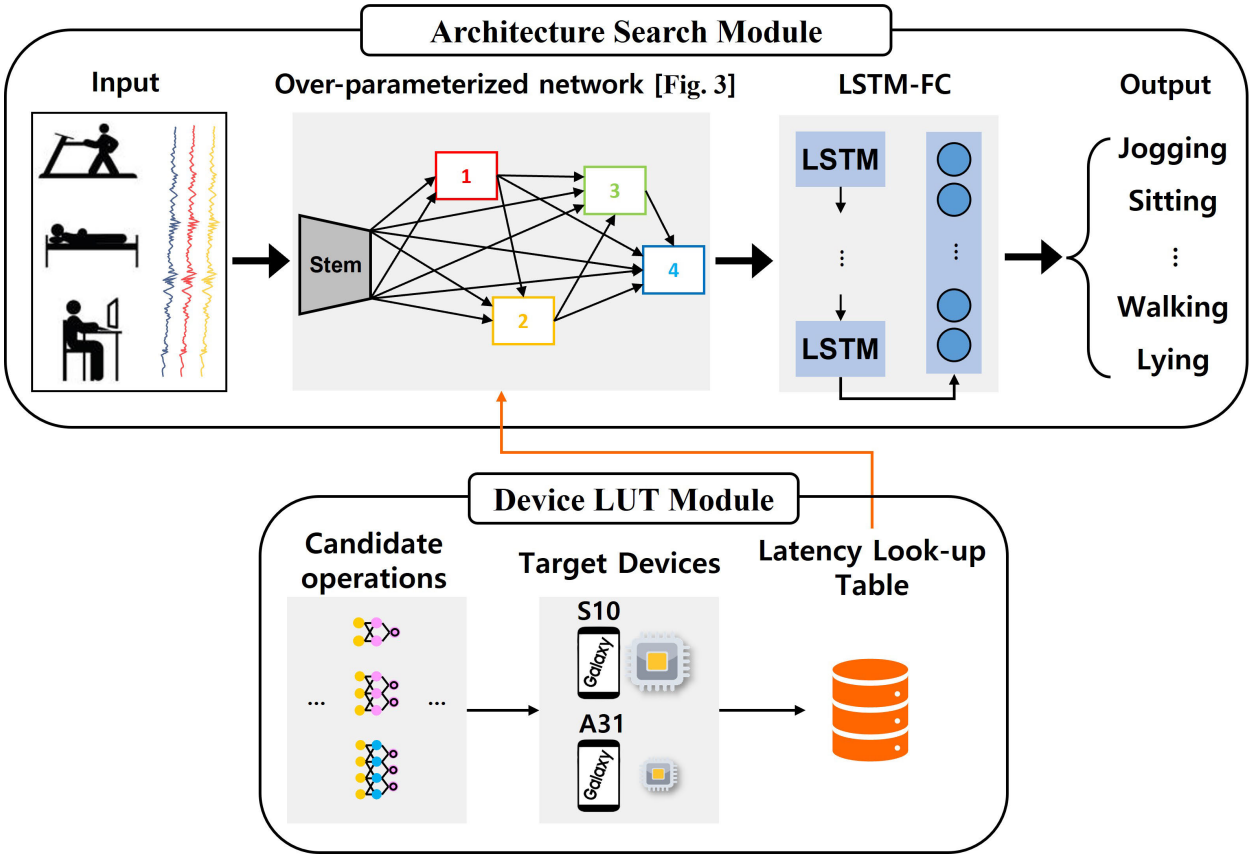


FIGURE 1. Mobile human activity recognition (HAR) neural architecture search (NAS) model consisting of the search and device look-up table (LUT) modules. The architecture search module (top) performs architecture search by training an overparameterized network that includes all possible architectures. In the over-parameterized network, the output feature of each node serves as the input to the LSTM-FC, which stands for Long Short-Term Memory with Fully Connected layers and predicts activities as its output. The device LUT module (bottom) plays a role in measuring and providing the latency on the device for the candidate operations required in the architecture search process in advance.

Only a few studies have been conducted to automate the CNN design process using neural architecture search (NAS) methods for human activity recognition [9], [10], [11], [12]. These studies are inspired by reinforcement learning and evolutionary algorithms, which are commonly employed in computer vision tasks [13], [14]. However, existing HAR NAS methods often produce less effective architectures by not considering the computational capacity of the target devices and requiring computationally expensive architecture searches.

To address these challenges, we propose a mobile HAR NAS approach based on differentiable NAS (DNAS), which incorporates the latency of real mobile devices during the architecture search process. The DNAS-based approach addresses [15] the computational expense associated with exploring discrete search spaces in traditional NAS approaches. To alleviate this issue, DNAS adopts a strategy of relaxing the search space into a continuous domain. This relaxation enables efficient optimization using gradient-based methods, allowing for faster convergence and exploration of architectural configurations. By using a single training process, the architecture search time of the proposed method was significantly reduced. Furthermore, the effectiveness of the mobile device optimization method was verified by deploying searched models on real smartphones.

This paper focuses on addressing the limitations of conventional NAS methods in real-life mobile HAR systems. The model's latency is important in capturing changes in human behavior using mobile devices. Each device has unique hardware specifications, requiring optimal architectures based on device characteristics. Conventional NAS methods suffer from high architecture search time and rely on indirect metrics instead of target device-specific optimization. To overcome these limitations, the proposed approach incorporates the latency measured on the target device into the objective function and facilitates fast search using a DNAS-based approach. The goal is to find an efficient HAR model that adapts quickly to different edge devices. By doing so, this approach offers an efficient and device-specific solution for mobile HAR NAS. The main contributions of this paper are summarized as follows:

1) We propose a mobile HAR neural architecture search based on a differentiable NAS (DNAS) that reflects the

latency of real mobile devices in the architecture search process.
2) By using a single training process, the architecture search time of the proposed method is significantly reduced compared to previous methods.
3) The effectiveness of the proposed method is verified by deploying searched models on real smartphones.

II. RELATED WORK

A typical choice for dealing with the time-series characteristics of HAR can be an RNN, which is capable of addressing the long-range time dependency of the given data [16]. In RNNs, a long short-term memory (LSTM) structure and gated recurrent unit (GRU) are used to capture the long-range dependency of sequence data. However, recent studies in HAR have focused on CNNs because they are capable of capturing human behaviors and hierarchically extracting features from low-level to high-level features through multiple convolutional layers. In DeepConvLSTM [6], CNN and LSTM structures of the RNN were integrated to achieve state-of-the-art performance on various public HAR datasets.

In conventional studies, deep learning architectures have been designed manually, which is difficult and time-consuming due to a series of factors involved in architecture design. These factors include operations, the number of layers, the connection between layers, the target device, and the recognition of activities. For example, the neural network architecture may become dense or loose depending on the complexity of the activity and the target device. Although designing a neural network architecture for various image recognition tasks has been automated using neural architecture search (NAS) [13], [14], [15], few studies have considered NAS for human activity recognition. To the best of our knowledge, there are two types of NAS methods, namely reinforcement learning (RL)-based NAS [11] and evolutionary algorithm (EA)-based NAS [12].

In these two methods, HAR neural architectures are used to achieve a high F1-Score with low floating point operations (FLOPs) or memory access cost (MAC). In the RL-based NAS, a meta-controller is trained, in which an optimal architecture is generated to provide a reward feedback such as F1-Score and FLOPs on HAR dataset to the meta-controller. In the EA-based NAS, NSGA-II [17], a multi-objective optimization method, is used to design a lightweight and fast neural architecture. Finally, the generated model is a neural architecture that achieves Pareto optimality for HAR tasks with respect to three objective metrics: F1-Score, FLOPs, and MAC.

III. MATERIALS AND METHOD

In this section, we explain the entire algorithm of the proposed method, from search space to search strategy and the way to reflect the latency of the target devices.

A. MOTIVATION

In HAR systems using a mobile device, the latency in the inference phase is important because quickly capturing changes in human behavior in fields such as fall detection for elderly people or rehabilitation exercise recognition for patients [18], [19] is critical. In particular, each mobile device exhibits distinct hardware specifications so that optimal architectures can differ depending on the latency of the mobile device. Conventional NAS methods exhibit two disadvantages when applied to real-life mobile HAR NAS. First, indirect metrics, such as FLOPs or MAC, are used instead of a metric directly related to the target device. Therefore, the target device-specific models may not be optimized because conventional methods exploit the indirect metrics during the NAS process; they do not optimize the actual latency from the target devices. Second, in previous HAR NAS studies, repeated architecture sampling, training, and evaluation are required, considerably increasing the architecture search time. For example, existing HAR NAS methods spend considerable time searching only one convolutional block architecture in the ConvLSTM baseline network. In the empirical tests, RL-based NAS [11] required approximately nine GPU hours on average for each of the well-known public HAR datasets, and EA-based NAS [12] required approximately seven GPU hours. The high architecture search time in these methods can be attributed to the retraining of the sampled architecture, which requires repeated sampling, training, and evaluation of the procedures.

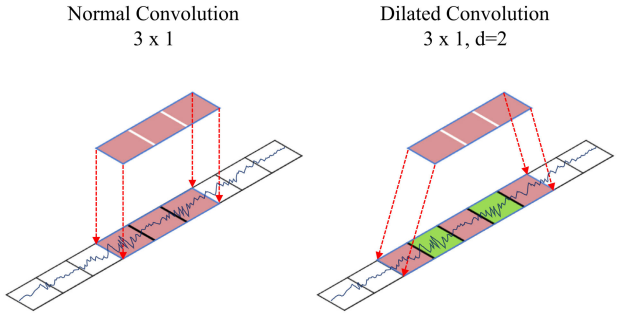


FIGURE 2. Normal Convolution (left), Dilated Convolution (right).

B. APPROACH

To address the limitations of conventional methods, we constructed an overparameterized network with a directed acyclic graph (DAG) structure that encompasses all possible architectures. We then conducted a DNAS-based architecture search, as depicted in the architecture search module of FIGURE 1. Utilizing the overparameterized network enables a quick search for the optimal architecture, as the network only needs to be trained once.

In order to directly optimize the latency for each mobile device instead of relying on FLOPs as an indirect measure of latency, we employed a novel approach. The latency used for optimization is measured in advance by the target mobile device, as shown in FIGURE 1, and stored in a latency lookup table (LUT). To generate the desired network, we utilized a loss function that combines a latency term as an objective metric with a cross-entropy term for learning accuracy.

TABLE 1. Type of operations in the search space: each operation consists of a pair of kernel, padding, and dilation values, and 13 candidate operations.

Operation	Kernel	Padding	Dilation
Convolution	(1,1)	(0,0)	-
Convolution	(3,1)	(1,0)	-
Convolution	(5,1)	(2,0)	-
Convolution	(7,1)	(3,0)	-
Convolution	(9,1)	(4,0)	-
Dilated Convolution	(3,1)	(2,0)	(2,1)
Dilated Convolution	(5,1)	(4,0)	(2,1)
Average Pooling	(3,1)	(1,0)	-
Average Pooling	(5,1)	(2,0)	-
Max Pooling	(3,1)	(1,0)	-
Max Pooling	(5,1)	(2,0)	-
Skip Connection	-	-	-
No Connection	-	-	-

FIGURE 1 provides an overview of our framework, which consists of two main modules. The first module is the architecture search module responsible for the search process, while the second module is the Device LUT module designed to incorporate latency information specific to the target device. The architecture search module includes a feature extractor network that inputs behavioral data and outputs useful features for prediction. Its primary role is finding the optimal network architecture for feature extraction. On the other hand, the Device LUT module measures the latency of each operation defined in the search space on the target device, storing the latency information in a lookup table. During the search process, the Device LUT module provides the stored latency information as needed.

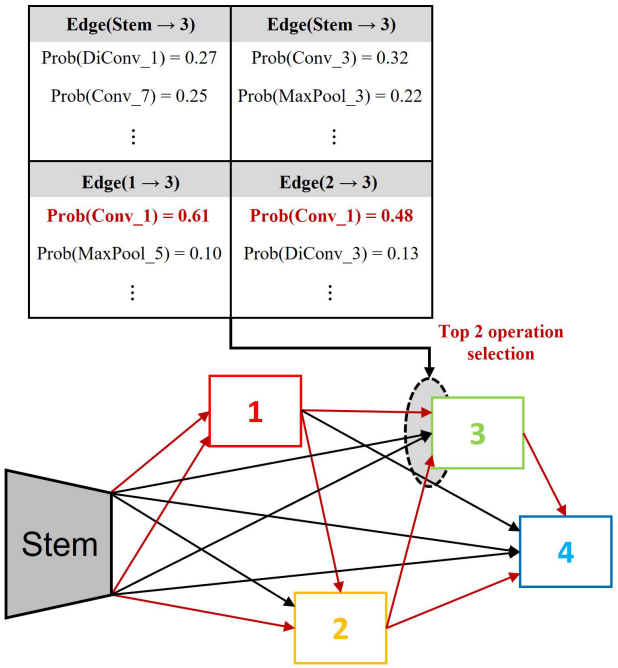


FIGURE 3. Overparameterized network with the search space. An overparameterized network is a DAG structure composed of nodes and edges. Each edge is calculated through a weighted sum that applies operations in the search space to the data. For instance, when the probability of each operation on the edge from node1 to node3 is provided, such as 0.61 for Conv_1, 0.10 for MaxPool_5, and so forth, the output of each operation is multiplied by its corresponding probability value. These multiplied outputs are then added together with the outputs of other operations, enabling each operation to contribute in proportion to its assigned probability value. Once the parameter update is complete, each node, such as node3, ultimately selects the two operations with the highest probability among all the edges.

C. SEARCH SPACE

To ensure the robust analysis of one-dimensional (1-D) time-series sensor data on human behavior, it is essential to design a search space with appropriate operations tailored for time-series data. In this regard, 1-D convolution is a suitable operation for extracting features from time-series data, as it applies the kernel in a single direction. Additionally, 1-D convolution can be categorized into two types of operations: normal and dilated, based on the dilation size. Despite having the same kernel size, the dilated convolution's kernel exhibits a wider receptive area than the normal convolution's kernel. As a result, the dilated convolution can extract more temporal features while requiring less computation [20]. For example, in FIGURE 2, the normal convolution recognizes a kernel area of 3×1. However, the dilated convolution expands the kernel by a dilation size of 2, resulting in a recognition area of 5×1. Considering this, our search space encompasses both normal convolutions with diverse kernel sizes of $\{1, 3, 5, 7, 9\}$ and dilated convolutions with kernel sizes of $\{3, 5\}$. Additionally, we incorporate average and max pooling operations for extracting higher-level features, skip connection operation to transmit information from previous layers to subsequent layers, and no connection operation to disconnect nodes. Therefore, as summarized in Table 1, the entire search space is primarily divided into six types of operations based on the "Operation" column, with each operation further categorized according to the kernel size. In the case of the convolution operation, the operation type is additionally subdivided based on the dilation size, resulting in a total search space of 14.

D. SEARCH STRATEGY

The high cost of architecture search time in previous HAR studies has been a bottleneck for modeling, as it involves repeatedly searching for models during the search process. To address this issue, we utilized the DNAS method [15] to design a mobile HAR neural network architecture. We denote an architecture space (S), in which we find an optimal architecture (arch $\in$ S) after training its weights (weight). We formulated the neural network architecture search problem as follows:

$$\min_{\text{arch} \in S} \min_{\text{weight}} L(\text{arch}, \text{weight}) \tag{1}$$

where L is a loss function to be defined. In order to employ the differential-based architecture search method, the discrete search space consisting of individual operations needs to be transformed into a continuous space. To achieve this,

we initially conducted a process called continuous relaxation. During this process, each operation is connected by architecture parameters, enabling a smooth transition within the search space. A set of candidate operations is expressed as $O = \{o^1, o^2, \ldots, o^M\}$, where M is the number of candidate operations ($M = 13$, as listed in Table 1). The architecture parameters corresponding to the candidate operations for continuous relaxation of this search space are $A = \{\alpha^1, \alpha^2, \ldots, \alpha^M\}$. All operations were connected by applying the softmax function to the architecture parameter α corresponding to each operation. As shown in the example of FIGURE 3, each edge is composed of a weighted sum of operations within the search space in Table 1. The connection of these edges forms a DAG structure, creating an overparameterized network. Therefore, in each edge, a mixed operation is represented as $\bar{o}$ and defined as follows:

$$\begin{aligned}\bar{o}(x) &= \sum_{i}^{M} \text{Prob}(i) \cdot o^i(x) \\ &= \sum_{i}^{M} \frac{\exp(\alpha^i)}{\sum_i^M \exp(\alpha^i)} \cdot o^i(x) \end{aligned} \tag{2}$$

where Prob(i) denotes the probability of selecting the ith operation, and $o^i(x)$ represents the parameter associated with the ith operation for the given input x. The probability Prob(i) is defined using the softmax function, where α^i represents the architecture parameter associated with the ith operation.

The architecture search process using an overparameterized network is summarized in Algorithm 1. Therefore, all edges are searched simultaneously by training architecture parameters α of the overparameterized network. Finally, the optimal architecture is generated by selecting the optimal operation with a high probability at each edge. Our search space has 13 candidate operations configured as presented in Table 1 for four nodes and each edge in the DAG. Our overall search space has 13 candidate operations for every 14 edges from two stem nodes to four nodes of the DAG; therefore, the possible architectures are $13^{14} \approx 4 \times 10^{15}$. The architecture search time is considerably reduced because these vast possible architectures are relaxed into one overparameterized network.

E. LATENCY METRIC OPTIMIZATION PROCEDURE

The loss function (1) should reflect not only the accuracy of the architecture but also the latency of the target device. Therefore, we define the following loss function:

$$\begin{aligned}&L(\text{arch, weight}) \\ &= \lambda \cdot \text{CE(arch, weight)} + (1-\lambda) \cdot \left[\frac{\mathbb{E}[\text{LAT}]}{\text{Target}}\right]^{\theta}\end{aligned} \tag{3}$$

where θ is the weight factor defined as follows:

$$\theta = \begin{cases} \beta, & \text{if } \mathbb{E}[\text{LAT}] \geq \text{Target} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

Algorithm 1 The Search Algorithm

```
input: Data_train, Data_val, O, A;
Create S with o ∈ O and α ∈ A
Initialize o and α in S
while L_val not converged do
    Freezing α
    Update weight by ∇_weight L_train(α, weight) with Data_train
    Freezing weight
    Update α by ∇_α L_val(α, weight) with Data_val
end while
output: S with trained parameters;
Extract the final architecture from S according to α
```

where β is the weight factor for $\mathbb{E}$[LAT] versus Target. The first term CE(arch, weight) denotes the cross-entropy loss of the architecture with the weight parameters. The second term $\mathbb{E}$[LAT] denotes the estimated latency of the architecture on the target device, where Target is the target latency specified by the user. Coefficient λ is a scaling factor that adjusts the scale of CE and $\mathbb{E}$[LAT] of the loss function, respectively. To calculate the $\mathbb{E}$[LAT] term, we used a latency LUT to estimate the overall latency of a network based on the actual latency of each operation [21]. The latency of each operation is measured on specific devices and stored in a LUT. This LUT contains the actual latency values for each operation on different target devices. During the search, the LUT is loaded based on the target device being considered, and the latency information is incorporated into the objective function. This allows the search algorithm to evaluate and optimize the architecture based on the measured latencies on the target device.

The overall latency of an overparameterized network was estimated by calculating the expectation values of each edge. Here, Prob(i, j) denotes the probability of selecting $o(i, j)$. lat($o(i, j)$) denotes the actual latency of $o(i, j)$ when loaded on the target device LUT of the ith edge and jth operation.

$$\mathbb{E}[\text{LAT}] = \sum_{i}^{N} \sum_{j}^{M} \text{Prob}(i, j) \cdot \text{lat}(o(i, j)) \tag{5}$$

Thus, we can estimate the overall latency of the overparameterized network.

IV. EXPERIMENTAL RESULTS

All the experimental results are composed of three parts. Section A. introduces the details of comparison methods, datasets, and implementations. Comparison results are discussed in Section B. For comparison of target devices, in-depth analysis is discussed in Section C.

A. EXPERIMENTAL SETUP

We implemented state-of-the-art methods to compare the performance against existing RL and EA-based methods, as well as our proposed method. Our experiments were

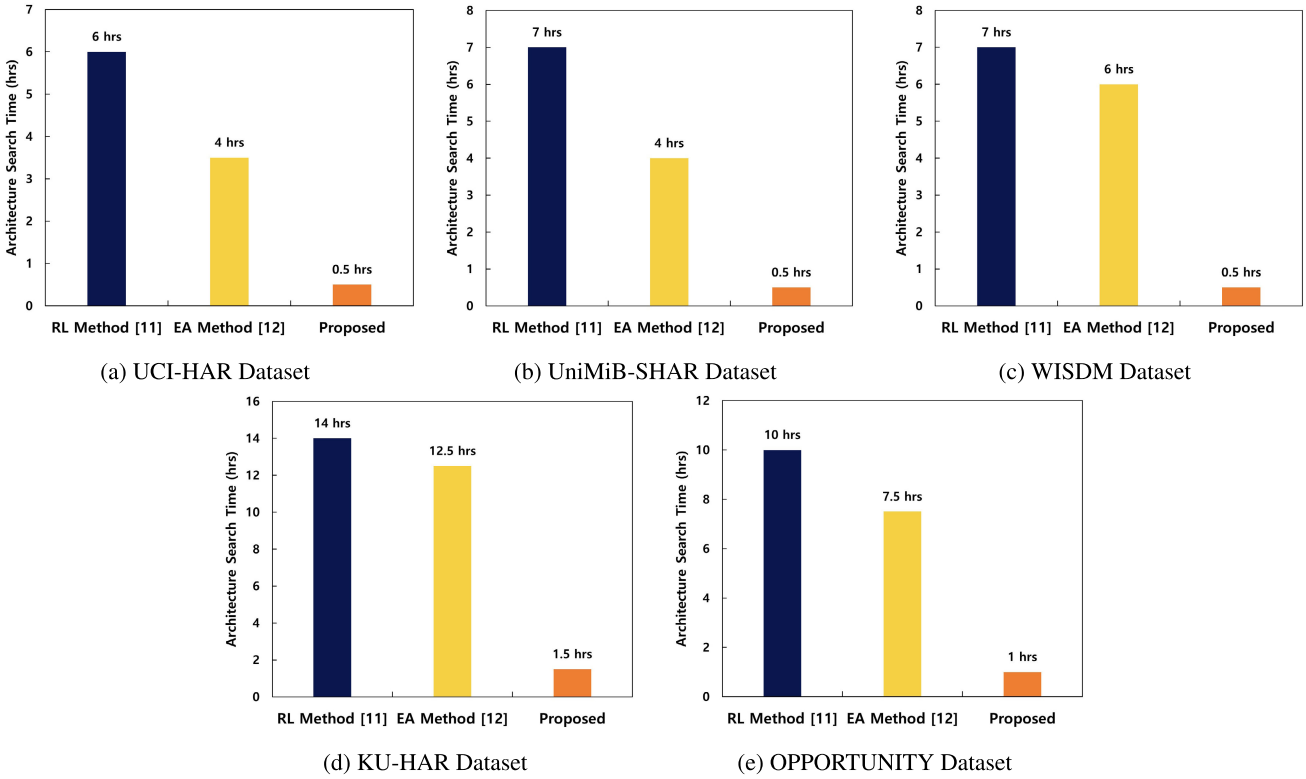


FIGURE 4. Comparison of architecture search time results for the proposed method and existing NAS methods [11], [12] on five HAR datasets.

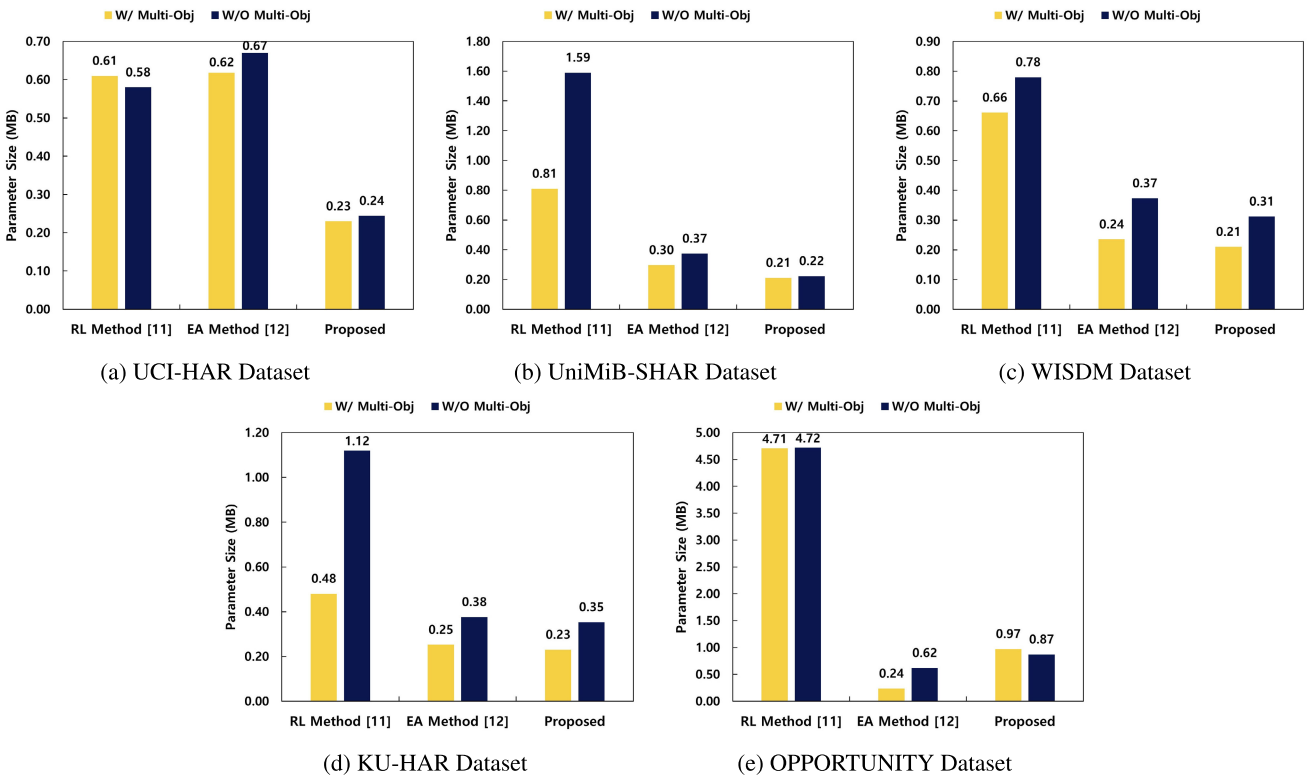


FIGURE 5. Comparison of parameter size results for the proposed method and existing NAS methods [11], [12] on five HAR datasets.

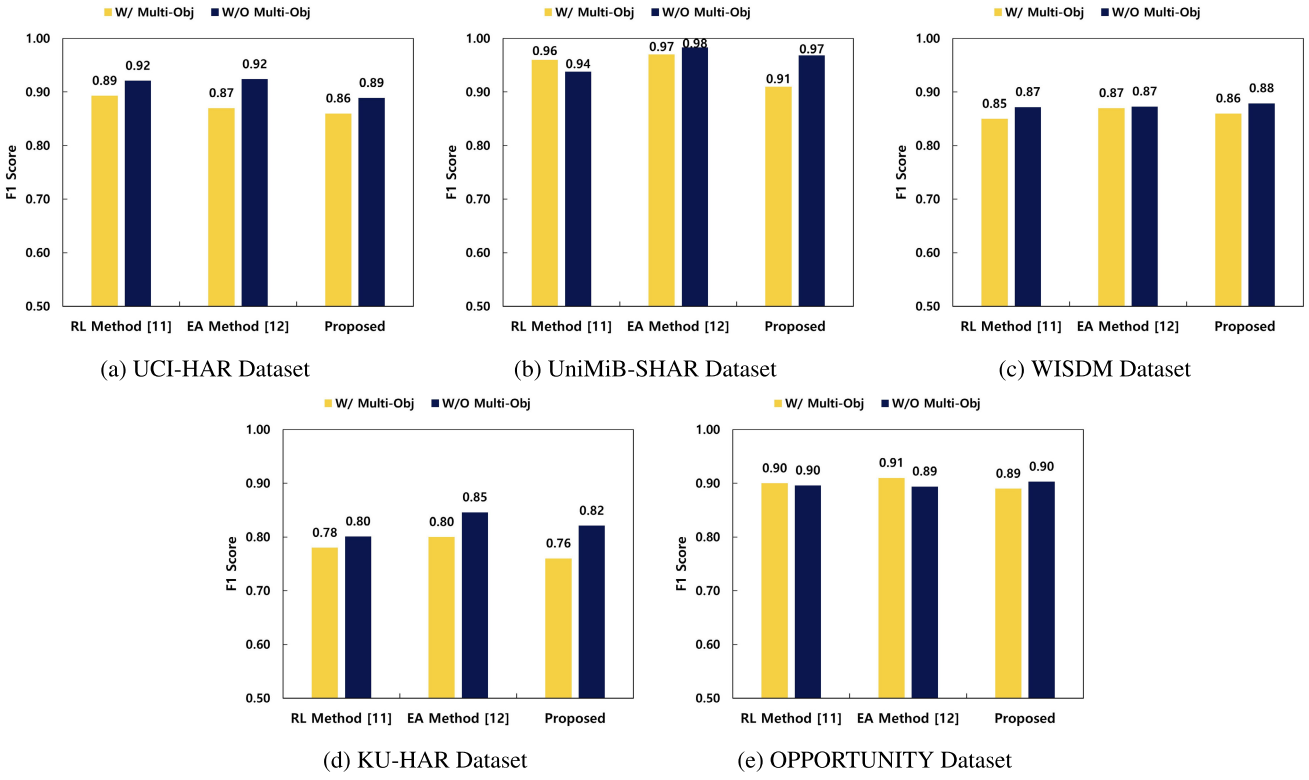


FIGURE 6. Comparison of F1-Score results for the proposed method and existing NAS methods [11], [12] on five HAR datasets.

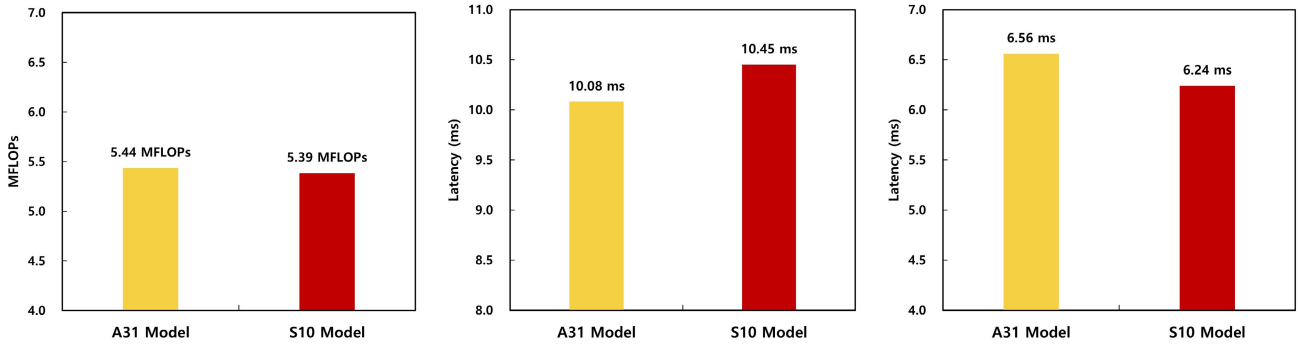
conducted on five public HAR datasets, where we measured the architecture search time, parameter size, and F1-Score as comparison metrics with existing methods. The architecture search time serves as an indicator of the search process efficiency, while the parameter size and F1-Score provide insights into the resulting architectures' size and accuracy, respectively. In addition to these evaluations, we conducted experiments targeting various target devices, including the Galaxy A31 and Galaxy S10, to assess the proposed method's optimization metric. Furthermore, to validate the effectiveness of our approach, we measured the latency on the Galaxy A31 with the Mediatek MT676 CPU and the Galaxy S10 with the Exynos 9820 CPU. In Section IV-B, we employed FLOPs as the optimization metric for the RL-based method, FLOPs and MAC for the EA-based method, and CPU latency for our proposed method. Furthermore, in Section IV-C, we specifically focused on the latency of the Galaxy A31 and Galaxy S10 devices as the optimization metric.

The five public HAR datasets used in experiments are as follows: UCIHAR [22], UniMiB-SHAR [23], WISDM [24], OPPORTUNITY [25], and KU-HAR [26]. The data of these datasets were collected by wearable sensors and mobile devices, which are widely used in HAR. The UCI-HAR dataset was collected from 30 subjects wearing a mobile device on their waist for six daily activities. The UniMiB-SHAR dataset was collected by recording 17 activities performed by 30 participants using acceleration sensors on mobile phones. The WISDM dataset was collected from 36 subjects wearing a mobile device on their pants and consisted of six daily activities. The opportunity dataset was collected by placing wearable devices on the upper body, buttocks, and legs of four subjects, who were then asked to perform 17 daily activities. The KU-HAR dataset was collected by recording 18 activities performed by 90 subjects by using acceleration and gyroscope sensors on their mobile phones.

We divided each dataset into a training set and test set as a proportion of 8:2, and the test set is also used to validate the overparameterized network. The overparameterized network was trained for 100 epochs with 256 batch sizes, and SGD and Adam were used as optimizers for learning the weights and architecture parameters, respectively. To balance the cross-entropy and latency term in the loss function (3), we set λ to 0.5, and β as 1 that are optimal setting values in our experiments. The framework used in the experiment was PyTorch deep learning library and was performed using an RTX 2080Ti GPU 11GB. Target device experiments were conducted on Android 10 OS systems.

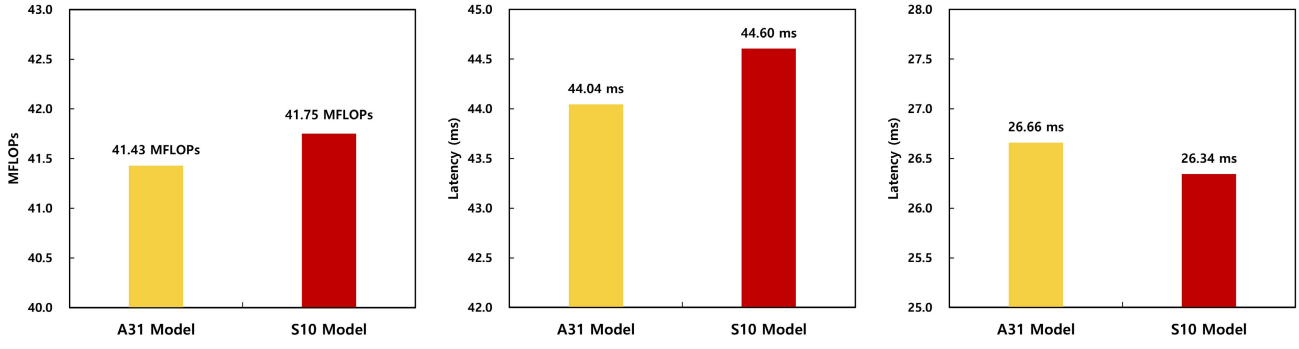
B. COMPARISON RESULTS

FIGURE 4 displays the architecture search times for each method using this dataset. As displayed in FIGURE 4 (a), the proposed method exhibited the lowest architecture search time (0.5 *hrs*, followed by 4 *hrs* for the EA method, and



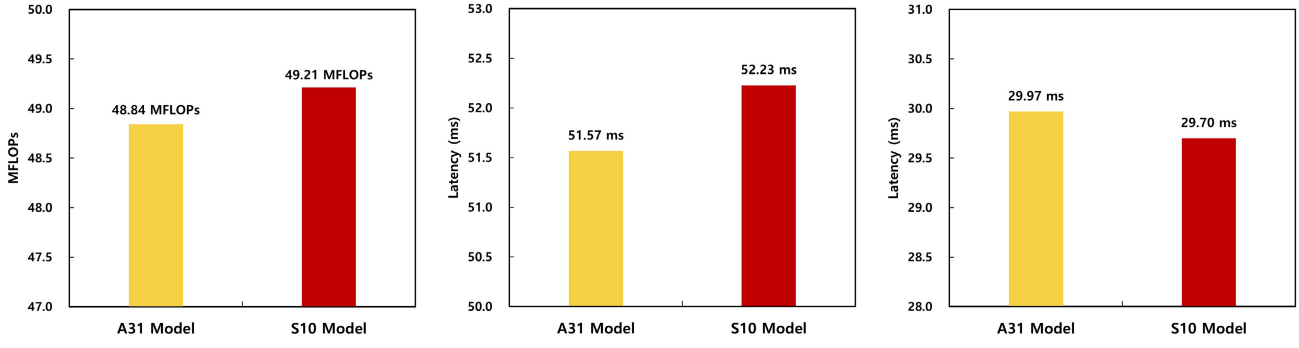
(a) FLOPs comparison (b) Latency on Galaxy A31 (c) Latency on Galaxy S10

FIGURE 7. FLOPs comparison with A31 and S10 model (a) and Latency comparison deployed on Galaxy A31 (b) and Galaxy S10 (c) for the models searched by the proposed method on the OPPORTUNITY dataset.



(a) FLOPs comparison (b) Latency on Galaxy A31 (c) Latency on Galaxy S10

FIGURE 8. FLOPs comparison with A31 and S10 model (a) and Latency comparison deployed on Galaxy A31 (b) and Galaxy S10 (c) for the models searched by the proposed method on the UCI-HAR dataset.

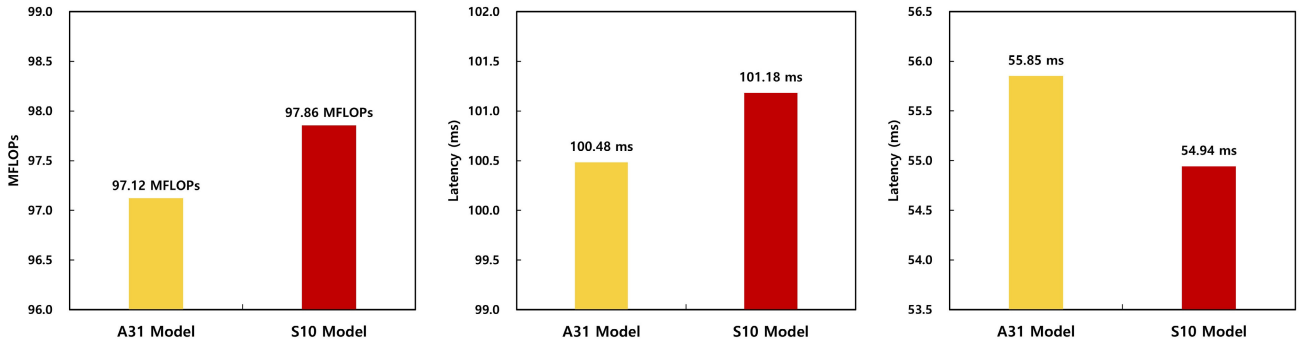


(a) FLOPs comparison (b) Latency on Galaxy A31 (c) Latency on Galaxy S10

FIGURE 9. FLOPs comparison with A31 and S10 model (a) and Latency comparison deployed on Galaxy A31 (b) and Galaxy S10 (c) for the models searched by the proposed method on the UniMiB-SHAR dataset.

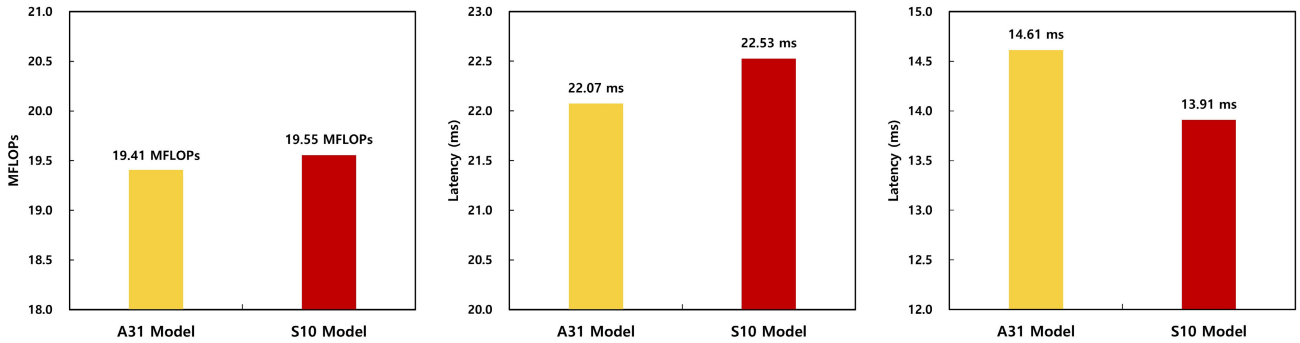
6 *hrs* for the RL method). Thus, in all five datasets of FIGURE 4, the proposed method exhibited the lowest architecture search time, and the RL method exhibited the highest architecture search time. The architecture search time of the proposed method is approximately (0.5−1.5 *hrs*), and it is 9−14 times faster than the RL method (6.0−14.0 *hrs*), and 7−12 times faster than the EA method (3.5−12.5 *hrs*).

FIGURE 5 and FIGURE 6 present the results based on whether the optimization metric was accompanied by multi-objective optimization. The "W/ Multi-Obj" category represents optimization with the multi-objective function set in each algorithm, while the "W/O Multi-Obj" category focuses solely on optimizing the accuracy performance metric.



(a) FLOPs comparison (b) Latency on Galaxy A31 (c) Latency on Galaxy S10

FIGURE 10. FLOPs comparison with A31 and S10 model (a) and Latency comparison deployed on Galaxy A31 (b) and Galaxy S10 (c) for the models searched by the proposed method on the KU-HAR dataset.



(a) FLOPs comparison (b) Latency on Galaxy A31 (c) Latency on Galaxy S10

FIGURE 11. FLOPs comparison with A31 and S10 model (a) and Latency comparison deployed on Galaxy A31 (b) and Galaxy S10 (c) for the models searched by the proposed method on the WISDM dataset.

FIGURE 5 displays the parameter size of the search architecture for each method using the dataset. Consideration of multi-objectives led to a decrease in parameter size in the majority of datasets. This reduction in parameter size was evident even in the results of the proposed method, which integrated the latency term, across four datasets. Despite having the shortest architecture search time, the proposed method consistently produced architectures with the smallest parameter size compared to the other two methods across most datasets. In particular, in FIGURE 5 (a), the parameter size of the RL and EA methods is 0.60 (MB), whereas the proposed method generates a model that is approximately three times lower with 0.23 (MB). These results revealed that creating a lightweight model instead of FLOPs or MACs is possible by optimizing latency.

FIGURE 6 displays the F1-Score of the searched architecture for each method by using the dataset. Without considering multi-objectives, the F1-score is generally high across different methods. Notably, in the OPPORTUNITY dataset shown in FIGURE 6 (e), the proposed method exhibited improved performance despite a reduction in parameter size. In contrast, the EA method encountered a decrease in performance despite an increase in parameter size. In FIGURE 6 (c), the F1-Score of the proposed method is 0.86, and the F1-Score of the RL and EA methods are 0.85 and 0.87, respectively. The proposed method has a similar F1-Score despite the much shorter architecture search time than the RL and EA methods. In the remaining datasets, the proposed method can search architectures with similar performance to the RL and EA methods with at least seven times less architecture search time.

C. IN-DEPTH ANALYSIS

In previous HAR NAS studies, HAR architecture models have been optimized for FLOPs. However, the objective of this study is to create an optimized HAR architecture model for each target device using the latency of the target device. To demonstrate that the architecture of an optimal model can vary depending on the specifications of the target device, we configured an entry-level device (Galaxy A31, CPU: Mediatek MT6768) and a high-end device (Galaxy S10, CPU: Exynos 9820). We measured the latency of operations in our search space on the Galaxy A31 and Galaxy S10 devices. To facilitate the optimization process, we created a latency LUT by storing the measured latencies in advance. In the proposed method, one of the objective functions includes a term related to latency. During the optimization

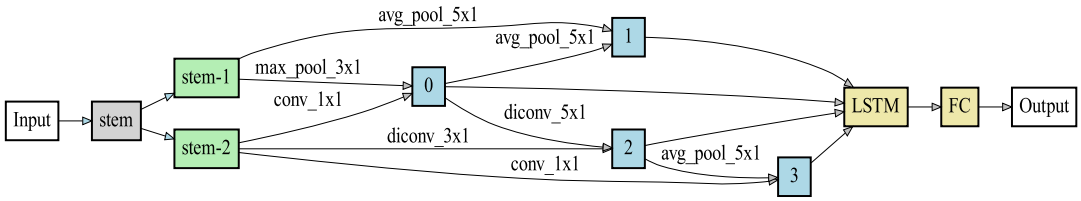


FIGURE 12. Block diagram illustrating the model for the Galaxy A31 on the OPPORTUNITY dataset.

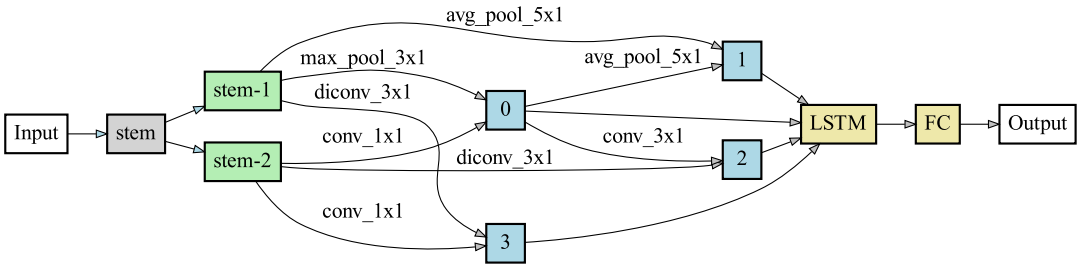


FIGURE 13. Block diagram illustrating the model for the Galaxy S10 on the OPPORTUNITY dataset.

process, we utilize the latency LUT to search and incorporate the latency of each operation specific to each device. By doing so, we can effectively consider and optimize for the latency characteristics of both devices in our approach. Since latency optimization depends on the device rather than the dataset, we utilized the latency LUT from the OPPORTUNITY dataset to perform the architecture search on Galaxy A31 and Galaxy S10. After the architecture search according to the target device, each model was deployed on both mobile devices for benchmarking the latency. Moreover, we revealed the portability of the obtained model by migrating it to the remaining four datasets. In FIGURE 7 to 11, (a) illustrates the comparison of FLOPs between the A31-optimized model and the S10-optimized model. (b) and (c) provide detailed comparison results of the latency for the A31-optimized and S10-optimized models on each respective device.

Comparing the FLOPs of each model in FIGURE 7 (a) revealed that the FLOPs of the S10-optimized model is 0.05 MFLOPs lower than that of the A31-optimized model. However, when the two models were deployed on the Galaxy A31 and Galaxy S10, the latency differed depending on the device. As displayed in FIGURE 7 (b), the latency of the A31-optimized model is 10.08 ms, which is lower than that of the S10-optimized model with fewer FLOPs, 10.45 ms. By contrast, in FIGURE 7 (c), the latency of the S10-optimized model was 6.24 ms, which is lower than that of the A31-optimized model of 6.56 ms. Experimental results revealed that the latency-based architecture search method proposed in this study is more appropriate for mobile device optimization than the FLOPs-based architecture search method. Likewise, similar results were observed when the model was migrated to other datasets. Although the S10 model has higher FLOPs in those datasets, the optimized model consistently shows lower latency on each respective device. Specifically, in the KU-HAR dataset experiment,

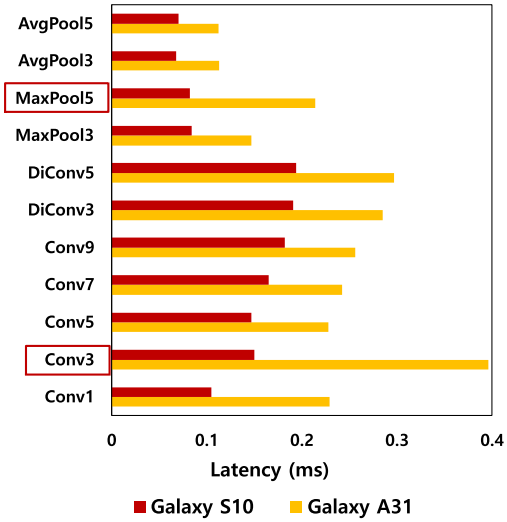


FIGURE 14. Comparison of operation latency on each mobile device(Galaxy A31, Galaxy S10) e.g., the latency of Conv3 has the most prominent distinction between devices.

FIGURE 10 (b) and (c) illustrate a significant latency difference of nearly 1 ms between the two models for both the Galaxy A31 and Galaxy S10.

Moreover, we conducted an investigation based on the latency LUT to analyze the differences between architectures using target mobile devices. We visualized the block diagrams of the model architectures optimized for each device in FIGURE 12 and FIGURE 13. Additionally, we depicted the latency LUT for candidate operations on Galaxy A31 and Galaxy S10 in FIGURE 14. A noticeable difference can be observed in the latency of each operation between Galaxy A31 and Galaxy S10. While Galaxy S10 benefits from ample

computational resources, Galaxy A31 exhibits more significant variations in operation latency. In particular, unlike Galaxy S10, Galaxy A31 displays high latency in convolution with a kernel size of three and max pooling with a kernel size of five. Conversely, Galaxy S10 exhibits lower latency in the convolution with a kernel size of three compared to other operations, and the difference in latency between each operation is smaller than that of Galaxy A31. By comparing and interpreting the latency of each operation on each device, it becomes evident that the model generated through the proposed method was tailored to reflect the characteristics of each device. FIGURE 12 shows that the convolution operation with a kernel size of three and the max pooling operation with a kernel size of five, which exhibited the highest latency on Galaxy A31, were not selected. Instead, operations with relatively lower latency were chosen. Conversely, as shown in FIGURE 13, on Galaxy S10, the convolution operation with a kernel size of three was selected due to its relatively low latency. This comparison highlights how the proposed method adapts the model architecture to the specific latency constraints of each device.

In summary, based on the additional experiments conducted, it is evident that the optimal model for HAR can vary across different devices due to variations in operation latency. Consequently, an architecture design that takes into direct consideration the hardware specifications of the target device was confirmed to be effective in environments with limited computational resources, such as smartphones.

V. CONCLUSION

In this study, we proposed a mobile HAR NAS based on a differentiable neural architecture search to automatically design the architecture of a HAR model for a mobile device. Our experiments achieved a significant milestone by utilizing a single training process incorporating an overparameterized network encompassing all candidate operations. This approach resulted in a remarkable reduction in search time on each dataset, with a duration of under 1.5 hours. As a result, the proposed method exhibited an average speed improvement exceeding sevenfold compared to conventional methods. Furthermore, our assessments on various target devices revealed that the proposed method, utilizing the target device's latency instead of focusing on FLOPS optimization like conventional HAR NAS methods, facilitates the exploration of hardware-specific architectures. In additional experiments conducted with the Galaxy A31 and Galaxy S10 smartphones as target devices, the latency of the A31-optimized model was on average 2% faster than the S10-optimized model on the A31 device. Specifically, when running the OPPORTUNITY dataset on the A31, the A31 model showed a 4% improvement in latency compared to the S10 model. On the other hand, when running the same dataset on the S10, the S10 model exhibited a 5% reduction in latency compared to the A31 model.

However, some concerns remain. The time and effort required to construct the latency LUT were not considered, which is not trivial if the number of devices and candidate operations increase. This problem could be alleviated by using a regression model trained from the latency LUT values to predict the latency of other devices.

REFERENCES

[1] F. Demrozi, R. Bacchin, S. Tamburin, M. Cristani, and G. Pravadelli, "Toward a wearable system for predicting freezing of gait in people affected by Parkinson's disease," *IEEE J. Biomed. Health Informat.*, vol. 24, no. 9, pp. 2444–2451, Sep. 2020.

[2] A. Prati, C. Shan, and K. I.-K. Wang, "Sensors, vision and networks: From video surveillance to activity recognition and health monitoring," *J. Ambient Intell. Smart Environ.*, vol. 11, no. 1, pp. 5–22, 2019.

[3] Y. Du, Y. Lim, and Y. Tan, "A novel human activity recognition and prediction in smart home based on interaction," *Sensors*, vol. 19, no. 20, pp. 4474–4489, 2019.

[4] P. Agarwal and M. Alam, "A lightweight deep learning model for human activity recognition on edge devices," *Proc. Comput. Sci.*, vol. 167, pp. 2364–2373, Jan. 2020.

[5] K. T. Chitty-Venkata and A. K. Somani, "Neural architecture search survey: A hardware perspective," *ACM Comput. Surv.*, vol. 55, no. 4, pp. 1–36, Apr. 2023.

[6] M. H. M. Noor, S. Y. Tan, and M. N. A. Wahab, "Deep temporal conv-LSTM for activity recognition," *Neural Process. Lett.*, vol. 54, no. 5, pp. 4027–4049, Oct. 2022.

[7] A. Bevilacqua, K. MacDonald, A. Rangarej, V. Widjaya, B. Caulfield, and T. Kechadi, "Human activity recognition with convolutional neural networks," in *Proc. Eur. Conf. ECML PKDD*, Dublin, Ireland, Sep. 2018, pp. 541–552.

[8] S. Mekruksavanich and A. Jitpattanakul, "RNN-based deep learning for physical activity recognition using smartwatch sensors: A case study of simple and complex activity recognition," *Math. Biosci. Eng.*, vol. 19, no. 6, pp. 5671–5698, 2022.

[9] W. N. Ismail, H. A. Alsalamah, M. M. Hassan, and E. Mohamed, "AUTO-HAR: An adaptive human activity recognition framework using an automated CNN architecture design," *Heliyon*, vol. 9, no. 2, Feb. 2023, Art. no. e13636.

[10] X. Wang, M. He, L. Yang, H. Wang, and Y. Zhong, "Human activity recognition based on an efficient neural architecture search framework using evolutionary multi-objective surrogate-assisted algorithms," *Electronics*, vol. 12, no. 1, pp. 50–69, 2022.

[11] L. Pellatt and D. Roggen, "Fast deep neural architecture search for wearable activity recognition by early prediction of converged performance," in *Proc. Int. Symp. Wearable Comput.*, Sep. 2021, pp. 1–6.

[12] X. Wang, X. Wang, T. Lv, L. Jin, and M. He, "HARNAS: Human activity recognition based on automatic neural architecture search using evolutionary algorithms," *Sensors*, vol. 21, no. 20, pp. 6927–6950, 2021.

[13] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8697–8710.

[14] Z. Lu, I. Whalen, V. Boddeti, Y. Dhebar, K. Deb, E. Goodman, and W. Banzhaf, "NSGA-Net: Neural architecture search using multi-objective genetic algorithm," in *Proc. Genet. Evol. Comput. Conf.*, Jul. 2019, pp. 419–427.

[15] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," 2018, *arXiv:1806.09055*.

[16] N. Dua, S. N. Singh, V. B. Semwal, and S. K. Challa, "Inception inspired CNN-GRU hybrid network for human activity recognition," *Multimedia Tools Appl.*, vol. 82, no. 4, pp. 5369–5403, Feb. 2023.

[17] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[18] L. Schrader, A. V. Toro, S. Konietzny, S. Rüping, B. Schäpers, M. Steinböck, C. Krewer, F. Müller, J. Güttler, and T. Bock, "Advanced sensing and human activity recognition in early intervention and rehabilitation of elderly people," *J. Population Ageing*, vol. 13, no. 2, pp. 139–165, Jun. 2020.

[19] W. Zhang, C. Su, and C. He, "Rehabilitation exercise recognition and evaluation based on smart sensors with deep learning framework," *IEEE Access*, vol. 8, pp. 77561–77571, 2020.

[20] R. Xi, M. Li, M. Hou, M. Fu, H. Qu, D. Liu, and C. R. Haruna, "Deep dilation on multimodality time series for human activity recognition," *IEEE Access*, vol. 6, pp. 53381–53396, 2018.
[21] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "FBNet: Hardware-aware efficient ConvNet design via differentiable neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 10726–10734.
[22] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones," in *Proc. 21st Int. Eur. Symp. Artif. Neural Netw.*, 2013, pp. 437–442.
[23] D. Micucci, M. Mobilio, and P. Napoletano, "UniMiB SHAR: A dataset for human activity recognition using acceleration data from smartphones," *Appl. Sci.*, vol. 7, no. 10, pp. 1101–1119, 2017.
[24] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *ACM SIGKDD Explor. Newslett.*, vol. 12, no. 2, pp. 74–82, Mar. 2011.
[25] D. Roggen, A. Calatroni, M. Rossi, T. Holleczek, K. Förster, G. Tröster, P. Lukowicz, D. Bannach, G. Pirkl, A. Ferscha, J. Doppler, C. Holzmann, M. Kurz, G. Holl, R. Chavarriaga, H. Sagha, H. Bayati, M. Creatura, and J. del R. Millàn, "Collecting complex activity datasets in highly rich networked sensor environments," in *Proc. 7th Int. Conf. Netw. Sens. Syst. (INSS)*, Jun. 2010, pp. 233–240.
[26] N. Sikder and A.-A. Nahid, "KU-HAR: An open dataset for heterogeneous human activity recognition," *Pattern Recognit. Lett.*, vol. 146, pp. 46–54, Jun. 2021.

WON-SEON LIM received the B.S. and M.S. degrees from Chung-Ang University, Seoul, South Korea, where he is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering. His research interests include continual learning, neural architecture search, and on-device AI.

WANGDUK SEO received the B.S. and M.S. degrees from Chung-Ang University, Seoul, South Korea, where he is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering. His research interests include feature selection, metaheuristic optimization, and neural architecture search.

DAE-WON KIM (Member, IEEE) received the B.S. degree from Kyungpook National University, Daegu, South Korea, and the M.S. and Ph.D. degrees from the Korea Advanced Institute of Science and Technology. He was a Postdoctoral Researcher with the Korea Advanced Institute of Science and Technology. He is currently a Professor with the School of Computer Science and Engineering, Chung-Ang University, Seoul, South Korea. His research interests include advanced data mining algorithms with innovative applications to bioinformatics, music emotion recognition, educational data mining, affective computing, and robot interaction.

JAESUNG LEE received the B.S., M.S., and Ph.D. degrees in computer science from Chung-Ang University, Seoul, Republic of Korea, in 2007, 2009, and 2013, respectively. He also studies classification, feature selection, and especially multilabel learning with information theory. He is currently the Head and an Associate Professor with the Department of Artificial Intelligence, Chung-Ang University. He is also the Chief of the AI/ML Innovation Research Center, Chung-Ang University. His research interests include machine learning, multilabel learning, model selection, and neural architecture search.