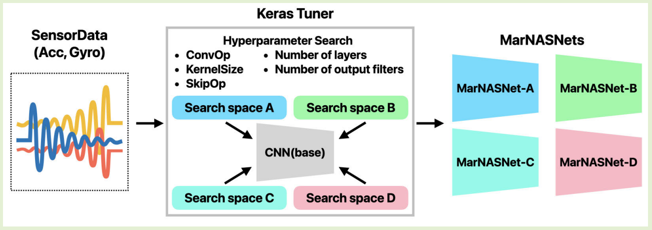
MarNASNets: Toward CNN Model Architectures Specific to Sensor-Based Human Activity Recognition

Satoshi Kobayashi, Tatsuhito Hasegawa, *Member, IEEE*, Takeru Miyoshi, and Makoto Koshino

***Abstract*—Deep learning (DL) models for sensor-based human activity recognition (HAR) are still in their nascent stages compared with image recognition. HAR's inference is generally implemented on edge devices such as smartphones because of its secure privacy. However, lightweight DL models for HAR, while meeting the hardware limitations, are lacking. In this study, using the neural architecture search (NAS), we investigated an effective DL model architecture that can be used for inference on smartphones. We designed multiple search spaces for the type of convolution, the kernel size of the convolution process, the type of skip operation, the number of layers, and the number of output filters by Bayesian optimization. We propose models called mobile-aware convolutional neural network (CNN) for sensor-based HAR by NAS (MarNASNets). We constructed four MarNASNet networks, MarNASNet-A to -D, each with a different model size and a parameter search space of four patterns. Experimental results show that MarNASNets achieve the same accuracy as the existing CNN architectures with fewer parameters and are effective model architectures for on-device and sensor-based HAR. We also developed Activitybench, an iOS app, for measuring model performance on smartphones, and evaluated the on-device performance of each model. MarNASNets' exploration achieved accuracy comparable to the existing CNN models with smaller model sizes. MarNASNet-C achieved accuracies of 92.60%, 94.52%, and 88.92% for Human Activity Sensing Consortium (HASC), UCI, and Wireless Sensor Data Mining (WISDM), respectively. Especially for HASC and UCI, MarNASNet-C achieved the highest accuracies despite the small model size. Their latency was also comparable to that of the existing CNN models, enabling real-time on-device inference.**



***Index Terms*—Convolutional neural network (CNN), deep learning (DL), human activity recognition (HAR), neural architecture search (NAS).**

I. Introduction

HUMAN activity recognition (HAR) is an important technology in healthcare and systems for monitoring the elderly. The recent spread of smartphones and smartwatches has simplified the use of motion sensors. Sensor-based HAR is being actively researched. For example, in the healthcare field, accelerometers have been used to track the amount of activity of obese people, patients, and so on [1], [2]. By recognizing human activity from sensor data, the time spent walking, running, bicycling, etc., can be automatically recorded, enabling more detailed healthcare support. HAR technology can also be applied to other products such as fall detection [3] and life logs [4], and the ability to recognize human activities in the real world can realize various valuable ubiquitous services.

To achieve high accuracy, sensor-based HAR typically employs machine learning (ML), particularly deep learning (DL). Previous HAR research has used knowledge-based techniques [5] and ML approaches, such as support vector machine (SVM) and decision tree, to classify activities by extracting statistical features from sensor data [6], [7]. However, in recent years, DL-based methods have been shown to provide higher estimation accuracy [8], [9], [10]. Despite the proposal of hybrid techniques that combine classical algorithms with

Manuscript received 2 June 2023; revised 2 July 2023; accepted 2 July 2023. Date of publication 11 July 2023; date of current version 15 August 2023. This work was supported in part by the Japan Society for the Promotion of Science (JSPS) KAKENHI Grant-in-Aid for Early-Career Scientists and Grant-in-Aid for Scientific Research (C) under Grant 19K20420 and Grant 23K11164. The associate editor coordinating the review of this article and approving it for publication was Prof. Subhas C. Mukhopadhyay. *(Corresponding author: Tatsuhito Hasegawa.)*

This work involved human subjects or animals in its research. The authors confirm that all human/animal subject research procedures and protocols are exempt from review board approval.

Satoshi Kobayashi and Tatsuhito Hasegawa are with the Graduate School of Engineering, University of Fukui, Fukui 910-8507, Japan (e-mail: t-hase@u-fukui.ac.jp).

Takeru Miyoshi is with the Department of Electronics and Information Engineering, National Institute of Technology, Ishikawa College, Tsubata 929-0392, Japan, and also with the Graduate School of Natural Science and Technology, Kanazawa University, Kanazawa 920-1192, Japan.

Makoto Koshino is with the Department of Electronics and Information Engineering, National Institute of Technology, Ishikawa College, Tsubata 929-0392, Japan.

Digital Object Identifier 10.1109/JSEN.2023.3292380

DL [11], the use of DL for HAR research has attracted significant interest due to its superior estimation accuracy. However, many studies using DL have adopted a shallow convolutional neural network (CNN) architecture [9], [12]. Deeper models [13] have been confined to the application of CNN models in the early stage of image classification, such as VGG [14], ResNet [15], Inception-v3 [16], and other models. It is unknown whether the latest models are suitable for sensor-based HAR. It is also unclear whether a CNN model architecture suitable for sensor-based HAR exists. DL generally requires considerable computational resources and mostly uses graphics processing units (GPUs) for high-speed processing. Therefore, services using DL models are typically provided on cloud servers equipped with GPUs. When using cloud services, data must be sent to the server, which is undesirable because of secure privacy. Therefore, inference processing using DL models should be performed on edge devices; however, devices such as smartphones have limited computational resources compared with servers. Therefore, a lightweight DL model that can run on smartphones at low latency is required. Based on this background, we propose models called mobile-aware CNN for sensor-based HAR by neural architecture search (MarNASNets), which construct effective models for sensor-based HAR through neural architecture search (NAS). MarNASNets aim to achieve higher accuracy with fewer parameters than the existing CNN model architectures, with the goal of constructing a highly accurate HAR model that runs at low latency on smartphones. The contributions of this research are as follows.

1) We proposed MarNASNets, CNN model architectures specialized for sensor-based HAR in terms of accuracy, latency, and model size (see Fig. 1). MarNASNets are implemented by searching for the optimal model architectures through NAS using Bayesian optimization. We also evaluated the accuracy of our model using three benchmark datasets, demonstrating that the accuracy of existing models can be achieved with more lightweight models.
2) We developed Activitybench, an iOS app, for measuring the on-device performance (CPU usage and latency) of HAR models on smartphones. We also evaluated MarNASNets on smartphones using Activitybench to assess its performance from a real-world perspective.

II. Related Studies

A. CNN Model Architectures

Various CNN models have been proposed in the field of image recognition. ImageNet [18] has been used as a benchmark dataset for image recognition. After the appearance of AlexNet [19] in 2012, which significantly surpassed existing methods using hand-crafted feature extraction and traditional ML at that time, various CNN models have been proposed to improve accuracy (GoogLeNet [20], Inception-v3 [16], VGG [14], ResNet [15], PyramidNet [21], Xception [22], DenseNet [23], and SENet [24]). Since 2015, CNN models that can run in environments with limited computing resources, such as smartphones, have also been proposed (MobileNets [25], MobileNetV2 [26], MobileNetV3 [27], etc.). Along with CNN models, various convolutional layers (regular convolution (Conv), depthwise separable convolution (SepConv) [22], and mobile inverted bottleneck convolution (MBConv) [26], [27], [30], modules, and techniques (inception module [16], skip connection [15], dense block [23], and squeeze-and-excitation block (SEBlock) [24]) have been proposed one after another. The typical CNN model architecture consists of only the target layer/module. Nonetheless, there is a possibility that a highly accurate and efficient model architecture can be constructed by successfully combining various layers and modules. However, at present, the methodology for determining such optimal combinations is still under study, particularly in the HAR field.

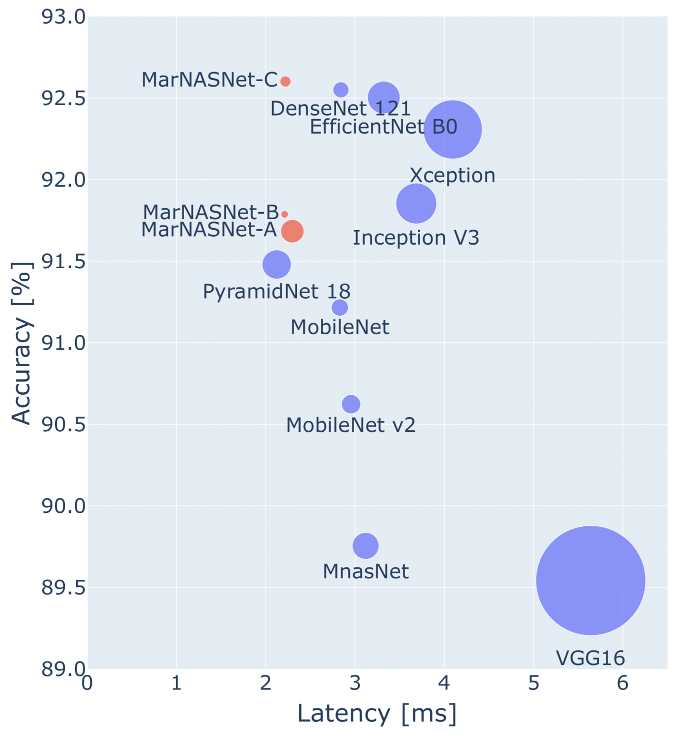


Fig. 1. Latency (processing time per one prediction) versus HASC [17] accuracy. Bubble size is directly proportional to the model size. Our MarNASNets achieve higher accuracy with small model size.

B. Neural Architecture Search

NAS is a method for automatically designing a model architecture by searching for the location, number, type, and output channel size of layers, modules, and techniques. NASNet [28] is a model architecture that consists of the results of a search for two cells, a "Normal Cell" and a "Reduction Cell," using reinforcement learning. MobileNetV3 and MnasNet [29] are mobile CNN models explored using NAS. In MnasNet, latency is measured using actual smartphones and is incorporated into the reward function of the search algorithm. MnasNet is 1.5× and 2.4× faster than MobileNetV2 and NASNet, respectively. EfficientNet [30] is a highly efficient model architecture that achieves high accuracy with a few parameters by scaling the model depth and breadth. EfficientNet-B1 to -B7 are constructed by searching the base EfficientNet-B0 with the same search space as MnasNet and adjusting the model scale. As shown above, the design of the model architecture, which has been performed manually, is now being performed by NAS to achieve high accuracy and efficiency.

C. Model Architecture in HAR

HAR is commonly implemented using CNNs, which have been successfully applied in the field of image recognition. The advantage of CNNs is that they can learn end-to-end feature representation and classification from data via backpropagation. Because features can be automatically extracted from data, it is possible to acquire effective features outside the range of human design. Because sensor data are time-series data, many researchers have employed recurrent neural networks (RNNs) or combined them with CNNs. Li et al. [9] employed CNN models with three alternating convolutional and pooling layers for sensor-based HAR. Similarly, Zeng et al. [12] proposed a HAR method from acceleration data using CNNs. Chen and Xue [31] applied a CNN and an SVM to sensor-based HAR and found that the CNN achieved higher accuracy. Han et al. [32] proposed a novel convolutional unit that uses convolutional layers with different kernel sizes. Huang et al. [33] pointed out a challenge arising from the different range of values for each channel when multiple sensors are used simultaneously in a multimodal context. To address this issue, they proposed channel normalization, which they subsequently integrated into a CNN and demonstrated its effectiveness. Ordóñez and Roggen [34] proposed a model (deep convolutional and long short-term memory (LSTM) RNNs: DeepConvLSTM) in which LSTM, a type of RNN, was inserted after a CNN and verified its effectiveness. Similarly, Xu et al. [13] proposed a model (InnoHAR) that combines an inception module proposed in the image recognition field and a gated recurrent unit (GRU), which is a type of RNN. Xia et al. [35] proposed a model that inserts LSTM in front of a CNN. Numerous other techniques have been proposed that combine CNN and RNN, such as a combination of CNN with GRU [36], a combination of CNN with hierarchically connected bidirectional LSTM (BiLSTM) [37], and a combination of SE-Nets, shortcut connections, and bidirectional GRU (BiGRU) [38]. Furthermore, several other approaches that combine CNNs and RNNs have been proposed [39], [40]. Debarshi et al. propose an ensemble model of CNN and LSTM structures. [41] Although many studies have applied DL to sensor-based HAR, they have been limited to simple CNN models and model architectures proposed in the early stages of image recognition, and many existing architectures are human-designed.

D. Position of This Study

Many studies have focused on the architectures of DL models that excel in specific domains, such as image recognition and natural language processing, where DL is the dominant application. In the field of HAR, several model architectures have been proposed, but most of them are merely adapted from image recognition architectures, and model architectures specific to HAR remain unexplored. The novelty of this research lies in elucidating a specialized DL model architecture for HAR, achieved through the exploration of NAS, which has not been explored previously. In addition, while previous studies have only evaluated performance on a personal computer, this study has developed a new iOS application called Activity-bench to evaluate on-device performance, which is a novel aspect of this research.

III. MarNASNets

We propose mobile-aware CNN for sensor-based HAR by NAS (MarNASNets), lightweight model architectures specialized for sensor-based HAR through Bayesian optimization using Keras Tuner.[1] Bayesian optimization considers the relationship between input x (hyperparameters) and output y (accuracy, etc.) as a black box function f and searches for the input x that maximizes or minimizes the output y, assuming that the function f follows a Gaussian process. While our previous work [42] only discussed the accuracy of various CNN models, the goal of this study is to discuss not only accuracy but also real-time inference on-device. Therefore, it is necessary to consider not only accuracy but also latency and CPU usage for the models.

A. Search Space

The model architecture to be explored consists of multiple convolutional blocks, which is common in various CNN model architectures. In this study, we adopted a search space in which each block individually searched for the type of convolutional layer, etc., referring to MnasNet. Fig. 2 shows a schematic of the search space used in this study. The search space consisted of four types (A/B/C/D) with different numbers of output filters and blocks. The space of the ith block is depicted below for each space.

1) *Search Space A:*
   a) *Convolutional Operations (ConvOp):* Conv, SepConv, and MBConv.
   b) *Convolutional Kernel Size (KernelSize):* 220 135.
   c) *Skip ops (SkipOp):* Pooling (pool), identity residual (identity), no skipping (none).
   d) *Number of Layers N_i:* 220 135.
   e) *Number of Output Filters F_i:* [16, 32, 64, 128].
2) *Search Space B:*
   a) *ConvOp:* Conv, SepConv, and MBConv.
   b) *KernelSize:* 220 135.
   c) *SkipOp:* Pool, identity, none.
   d) *Number of Layers N_i:* 220 135.
   e) *Number of Output Filters F_i:* 322 013 128.
3) *Search Space C:*
   a) *ConvOp:* Conv, SepConv, and MBConv.
   b) *KernelSize:* 220 135.
   c) *SkipOp:* Pool, identity, none.
   d) *Number of Layers N_i:* 220 135.
   e) *Number of Output Filters F_i:* 322 013 192.
4) *Search Space D:*
   a) *ConvOp:* Conv, SepConv, MBConv, and extreme inception module.
   b) *KernelSize:* 220 135.

[1]https://github.com/keras-team/keras-tuner

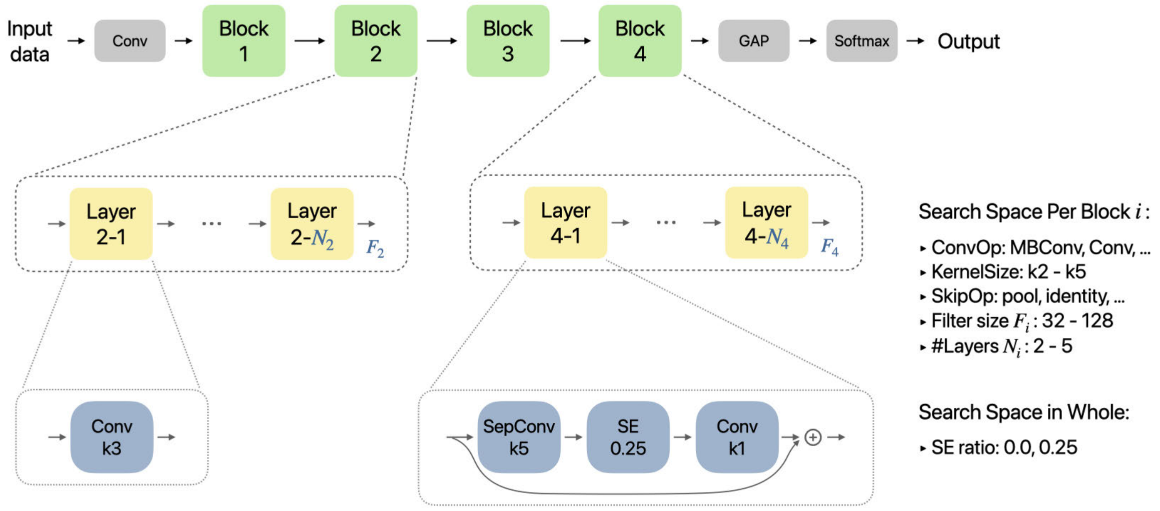


Fig. 2. Overview of search space. The top row indicates the overall model architecture; each block is investigated based on the search space.

c) *SkipOp:* Pool, identity, none.
d) *Number of Layers N_i:* 220 135.
e) *Number of Output Filters F_i:* 32–320.

ConvOp, *KernelSize*, *SkipOp*, and F_i determine the layer architecture, and N_i is a parameter that determines the number of times a conv layer is repeated in a block. For example, each layer of Block 4 in Fig. 2 is MBConv, which is repeated N_4 times, and the *number of output filters* is F_4. A common parameter explored is the squeeze-and-excitation [24] ratio: ***SERatio*** (0.0, 0.25). The number of convolutional blocks is fixed at 4 for search spaces A/B/C, and from 3 to 5 for search space D. Modern architectures proposed for image recognition consist of many convolutional blocks, and MnasNet also consists of seven convolutional blocks. However, to reduce the model size, we limited the number of convolutional blocks. For the *number of output filters*, the search is performed in such a way that four values are selected for search space A, and for the other search spaces, values are searched for in increments of four from the specified range.

B. Search Settings

When searching for the model architecture, we aimed to reduce the number of parameters as well as inference performance. The following blocks are used in the search: a Conv block, a SepConv block proposed by MobileNets, and a MBConv block proposed by MobileNetV2 and used in EfficientNet. The accuracy of Inception-v3 was high in the previous study [42], therefore, we expect that the accuracy will be high by using the inception block. However, because of concerns regarding the large size of the target model, we did not include it in the search space.

C. Evaluation of CNN Models on Smartphones

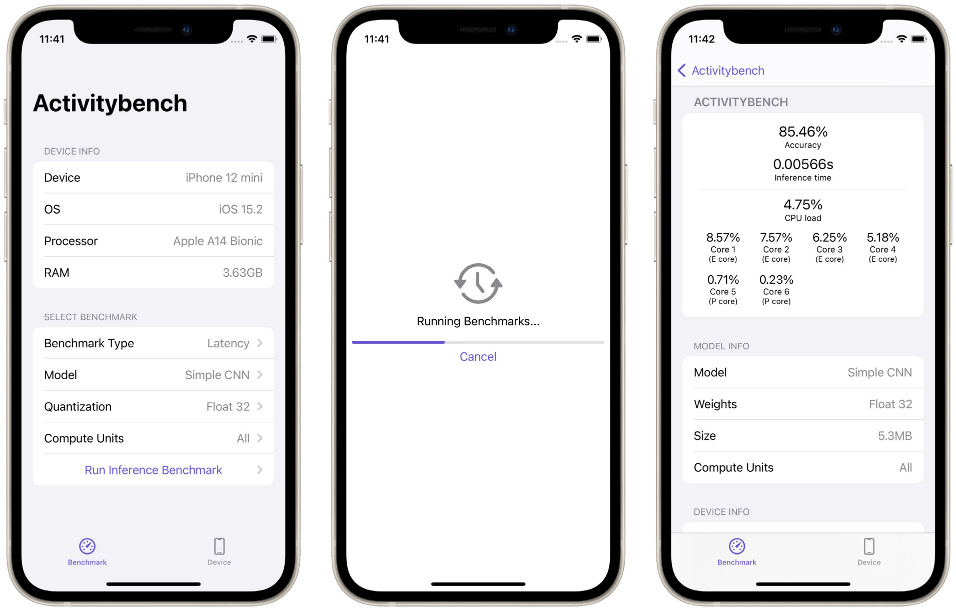
1) Model Transformation: We implemented an application that runs the CNN model on a smartphone and measured the latency and CPU usage of the model in an actual environment. Core ML is an ML framework developed by Apple, and optimized for Apple's hardware. It allows developers to run ML models completely on-device. In this study, models implemented in TensorFlow were converted to ML Model format using Core ML Tools and used in Core ML.

2) Activitybench: We describe the iOS app "Activitybench" developed to run models on the iPhone. Fig. 3 displays screenshots of Activitybench. Selecting a model on the screen shown in Fig. 3 (left) and tapping "Run Inference Benchmark" starts the measurement of latency and CPU usage. Fig. 3 (center) depicts the screen during execution, and Fig. 3 (right) depicts the screen displaying the results after execution. The results are sent to the Firestore Database. Activitybench collects acceleration data at 0.01-s intervals (100 Hz) using the Core Motion framework, and inference is performed after three-axis 256 samples are collected in 1 min. The average latency is the average time taken for each inference performed in 1 min. The average CPU usage is obtained after each inference is performed. CPU usage is measured by both the overall usage rate and usage rate of each core.

IV. Experiments

A. Datasets

In this study, we used three sensor-based HAR datasets: Human Activity Sensing Consortium (HASC) [17], UCI HAR using smartphones dataset (UCI) [43], and Wireless Sensor Data Mining (WISDM) [44], which were publicly available on the web. Details of each dataset are listed in Table I. HASC used the acceleration data of 170 individuals collected by an iOS device at a sampling frequency of 100 Hz. From each measurement data, time-series segmentation was performed with a window size of 256 samples and a stride of 256 samples as a preprocessing step. 5 s before and after the start of the measurement were trimmed to remove the influence of the device's storing behavior and other factors. UCI was measured at a sampling frequency of 50 Hz, and each measurement data sample is provided in a window of 128 samples. UCI provides two types of acceleration data, but total acceleration was used in this study. The gyroscope data were also used. Acceleration and gyroscope data were concatenated in channel direction, and six-axis sensor data were used as inputs. WISDM was measured at a sampling frequency of 20 Hz. As preprocessing,



Activitybench

DEVICE INFO

Device	iPhone 12 mini
OS	iOS 15.2
Processor	Apple A14 Bionic
RAM	3.63GB

SELECT BENCHMARK

Benchmark Type	Latency
Model	Simple CNN
Quantization	Float 32
Compute Units	All

Run Inference Benchmark

Running Benchmarks...

ACTIVITYBENCH

85.46%
Accuracy
0.00566s
Inference time

4.75%
CPU load

8.57% Core 1 (E core)	7.57% Core 2 (E core)	6.25% Core 3 (E core)	5.18% Core 4 (E core)
0.71% Core 5 (P core)	0.23% Core 6 (P core)		

MODEL INFO

Model	Simple CNN
Weights	Float 32
Size	5.3MB
Compute Units	All

DEVICE INFO

Fig. 3. Screenshots of iOS app: Activitybench, which uses HAR models on the iPhone to measure latency and CPU usage.

TABLE I
DETAILS OF DATASETS THAT WE USED IN EXPERIMENTS

Name	Num. of subjects in $D_{train} : D_{valid} : D_{test}$	Sampling frequency	Input shape	Output shape	Activities
HASC [17]	100 : 50 : 20	100 Hz	(?, 256, 3)	(?, 6)	stay, walk, jog, skip, stUp, stDown
UCI [43]	16 : 5 : 9	50 Hz	(?, 128, 6)	(?, 6)	walking, walking upstairs, walking downstairs, sitting, standing, laying
WISDM [44]	25 : 5 : 6	20 Hz	(?, 256, 3)	(?, 6)	Walking, Jogging, Upstairs, Downstairs, Sitting, Standing

we trimmed 3 s before and after each measured dataset and divided the time-series into 256 window size samples and 256 stride samples. Each dataset was divided into training (D_{train}), validation (D_{valid}), and test data (D_{test}). The subject-based hold-out method was used to divide the dataset into training, validation, and test data, with the subject as the as indicated in Table I.

B. Experimental Setup

We used HASC for the network-searching process. For comparison, we used the data of 170 subjects in HASC, as in the previous study [42], split into the training data (D_{train}), validation data (D_{valid}), and test data (D_{test}) in the ratio of 100:50:20 (subject-based hold-out method). D_{train} and D_{valid} are used for the search.

The objective function is the accuracy of classifying human activity for D_{valid}. In searching, the network was optimized by maximizing the objective function.

From the preliminary experiments, the number of searches was set to 1000 times, and the number of epochs in each search was set to 20. Analysis of the learning curve in the preliminary experiments showed that the learning curve generally converged around 20 epochs; therefore, the number of epochs was set to 20 to reduce the search time. The other hyperparameters were as follows.

1) *Batch Size:* 1024.
2) *Optimizer:* Adam.
3) *Learning Rate:* 0.001.
4) *Data Augmentation:* Flipping and channel-shuffling for D_{train}.

The model architectures were explored on a computer with an Intel Core i9-9900X, 64 GB RAM, and NVIDIA TITAN RTX.

C. Searched Networks

Fig. 4 provides an overview of the MarNASNet-A/B/C/D model, which is obtained by the search spaces A/B/C/D. In this section, we analyze which option is more likely to be selected within the set search space likely to be selected through the analysis of search C. The common architecture of the four MarNASNets is as follows.

1) *KernelSize:* 5.
2) *ConvOp:* MBConv (see Fig. 5).
3) *SkipOp:* Identity (see Fig. 6).
4) *Number of Output Filters:* Minimum or maximum value for each search range (see Fig. 7).
5) *SERatio:* 0.25 (=SE block is inserted, Fig. 8).

This shows that the SE block, skip connection (identity residual), and MBConv are also effective methods for sensor-based HAR.

Although a kernel size of 5 is typically chosen, all MarNASNets are not composed of only layers with a kernel size of 5. The diversity of kernel sizes is also considered an important element in the model architecture. (We discuss the diversity of kernel size and ConvOp in depth in Section V-A.)

Fig. 5 illustrates the number of times each *ConvOp* option is selected in each of the four blocks as a bar chart. In Search C, MBConv is most likely to be selected in all blocks, except Block 4, and Conv is most likely to be selected in Block 4.

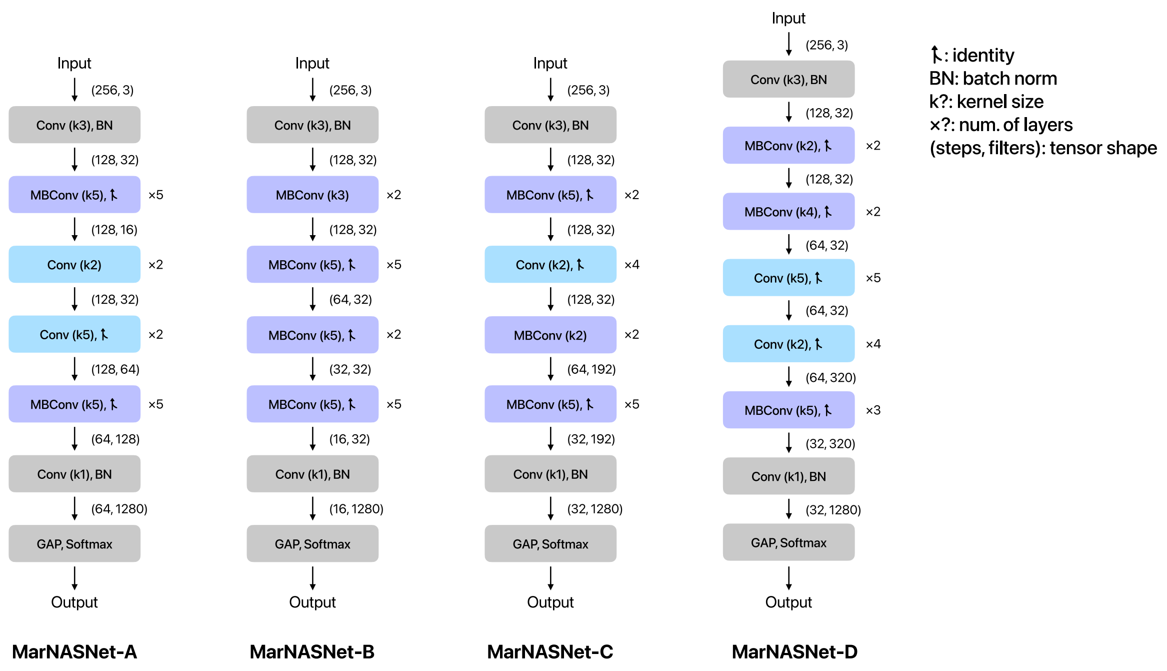


Fig. 4. MarNASNets model overview. MarNASNet-A to -D is found by NAS based on the search space A–D.

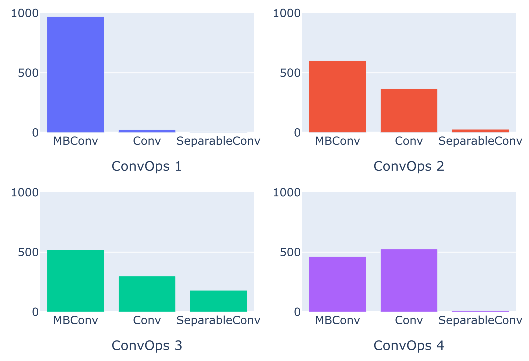


Fig. 5. Number of times each *ConvOp* option is selected in search space C.

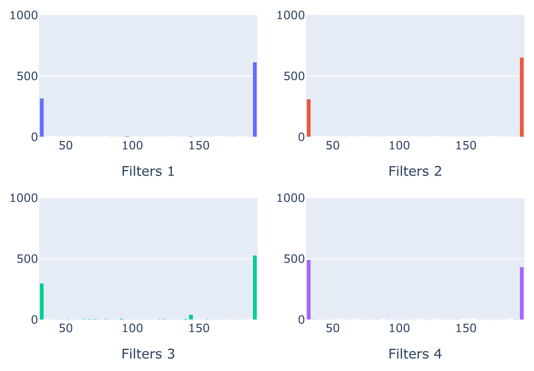


Fig. 7. Number of times each choice for the *number of output filters* F_i is selected in search space C.

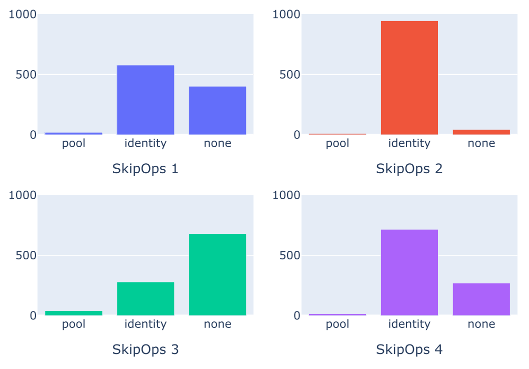


Fig. 6. Number of times each *SkipOp* option is selected in search space C.

Fig. 6 illustrates the number of times each *SkipOp* option is selected for each of the four blocks as a bar chart. The skip connection (identity residual) is most likely to be selected in search C.

Fig. 7 illustrates a bar chart of the number of times each option for the *number of output filters* F_i is selected for each of the four blocks. The minimum and maximum values of the search space are mostly chosen, and no intermediate values are chosen. This may be because too many search combinations are not selected in this search setup.

Fig. 8 illustrates the number of times each option in *SERatio* is selected as a bar chart. In search C, the majority of the choices are to insert SE blocks. A similar trend was observed for other search spaces. These results indicate that the search process tended to select many options, resulting in an architecture similar to EfficientNet. The model architecture that achieved the best accuracy in the search, shown in Fig. 4 also has an architecture similar to EfficientNet. The TensorFlow implementation of MarNASNets is available on GitHub.[2]

D. Experimental Setup for Evaluating Model Performance

We evaluated the effectiveness of the proposed MarNASNet-A to -D in terms of accuracy, latency, and

[2]https://github.com/Shakshi3104/tfmars

TABLE II
PERFORMANCE OF MARNASNETS AND OTHER CNN MODELS

Model	Accuracy [%] ⇑			Size [MB] ⇓	Latency [ms] ⇓	CPU [%] ⇓
	HASC	UCI	WISDM			
Simple CNN [9]	87.71	89.98	89.55	5.31	4.37	1.59
VGG16	89.54	90.40	89.32	154.00	5.64	1.83
Inception-v3	91.85	**94.23**	**91.54**	57.22	3.69	1.51
ResNet 18	90.53	91.44	87.53	15.41	2.67	1.50
PyramidNet 18	91.48	92.56	85.23	**1.63**	**2.12**	1.65
Xception	92.31	92.32	90.17	82.69	4.09	2.10
DenseNet 121	**92.55**	92.52	88.75	22.31	2.84	2.11
MobileNet	91.22	80.32	88.82	23.96	2.83	1.88
MobileNetV2	90.62	81.62	74.71	26.91	2.96	1.61
MobileNetV3 Small	91.45	91.25	82.47	11.60	2.48	**1.42**
NASNet Mobile	86.46	54.96	84.98	16.55	3.23	2.65
MnasNet	89.75	87.43	84.57	37.44	3.12	1.66
EfficientNet B0	**92.50**	93.53	89.11	45.70	3.32	1.59
EfficientNet lite0	91.52	91.20	85.81	43.11	3.21	1.89
MarNASNet-A	91.68	92.15	88.52	**1.31**	2.30	1.68
MarNASNet-B	91.79	94.20	**90.62**	**0.42**	**2.21**	**1.47**
MarNASNet-C	**92.60**	**94.52**	88.92	3.08	**2.22**	1.83
MarNASNet-D	91.70	**95.33**	90.34	8.16	2.86	**1.46**

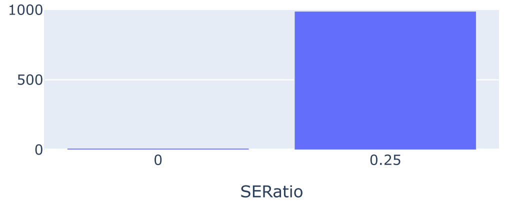


Fig. 8. Number of times each *SERatio* option is selected in search space C.

CPU usage. Accuracy was evaluated using benchmark datasets (HASC, UCI, and WISDM). Hyperparameters for model training were the same as in Section IV-B. Latency and CPU usage were evaluated using Activitybench on an iPhone 12 mini running iOS 15.2. Accuracy was evaluated as the average accuracy of five trials. Subjects were randomly sampled for each trial; for UCI only, nine standard-defined test users were fixed as D_{test}, and only D_{train} and D_{valid} were randomly sampled for each trial.

E. Experimental Results on Model Performance Evaluation

Table II presents the accuracy, model size, latency, and overall CPU usage of MarNASNets and models evaluated in previous studies [42] on the three datasets. For HASC, MarNASNets-C achieves an accuracy similar to DenseNet121 and EfficientNet-B0, whereas MarNASNet-D achieves the highest accuracy in UCI. In WISDM, MarNASNet-B achieved good accuracy even though it was not as good as that achieved by Inception-v3. In this study, we set the accuracy of the HASC validation data as an objective function of NAS, but MarNASNets also construct robust and accurate models on other datasets such as UCI and WISDM. Next, we focus on the model size: MarNASNet-A and -B have smaller model sizes than PyramidNet18, the smallest model of all the existing CNN models, MarNASNet-C and -D also have larger model sizes than MarNASNet-A and -B but smaller model sizes than the existing CNN models. The latency and CPU usage of our model are comparable to the existing CNN models. These results show that MarNASNets achieve the same accuracy as the existing CNN models with a smaller model size (i.e., fewer parameters), and their latency and CPU usage are sufficient to enable real-time inference on the device.

In this study, we used three different datasets: HASC (100 Hz - 256 samples per window), UCI (50 Hz - 128 samples per window), and WISDM (20 Hz - 256 samples per window,) as input data. Inputs of HASC and UCI are composed of 2.56 s, whereas those of WISDM are composed of approximately 10 s according to the previous study [45]. Although it is known that the longer the input series length, the higher the estimation accuracy in HAR, the estimation accuracy may not have been improved in WISDM compared to other datasets. We considered that one reason is that the MarNASNets are optimized based on the HASC validation accuracy. Therefore, it has a possibility to improve estimation accuracy in WISDM by searching the MarNASNet architecture based on other reward functions.

V. DISCUSSIONS

MarNASNet-A to D were obtained by exploring the model architectures using NAS. In this section, we evaluate the performance of MarNASNet-C, the most well-balanced of the obtained model architectures, by varying the number of convolution layers and the number of filters.

A. Diversity of Convolution Layers

In Section IV, we found the model architecture comprising an irregular mixture of various elements, such as MBConv–Conv and k2–k5. To investigate their influence of them, we conducted an ablation study in the mixed and unified cases for each factor. In this study, we validated MarNASNet-C and compared its accuracy with slightly modified MarNASNet-C models. The four model architectures were validated using HASC as the dataset.

Table III shows the accuracy of MarNASNet-C and the four validated models on HASC. MarNASNet-C achieves the best accuracy compared with the model architectures with a fixed

TABLE III
ACCURACY OF MARNASNET-C AND ITS VARIANTS ON HASC

Model	***KernelSize***	***ConvOp***	**Accuracy** [%] Mean (SD)
MarNASNet-C	mix	mix	**92.60** (1.94)
k5 only	5	mix	91.65 (1.74)
k2 only	2	mix	90.18 (2.93)
MBConv only	mix	MBConv	92.13 (2.19)
Conv only	mix	Conv	88.95 (2.30)

kernel size and convolution layer type. According to this result, a variety of *ConvOp* and *KernelSize* values positively impact accuracy. Regarding the effects of each factor, "k5 only" achieves a higher accuracy than "k2 only" about *KernelSize*, and "MBConv only" achieves a drastically higher accuracy than "Conv only" about *ConvOp*. These trends are similar to those of the MarNASNet architectures because MarNASNets frequently selected MBConv with a kernel size of 5.

B. Search Space for the Number of Filters

MarNASNets achieved the same accuracy as the existing CNN models with fewer parameters. However, Fig. 7 shows that, in most cases, the minimum or maximum value of the search space is selected for the *number of output filters*. One reason for this may be that the maximum *number of output filters* is 192 and the minimum is 32, which is a wide range, and that the search space is too wide, with four increments in the range of 322 013 192. We checked how the selection of output filters changes by narrowing the search space of the *number of output filters* and reducing the number of choices. According to the MnasNet [29], the size of the search space $\mathcal{S}$ is calculated as $\mathcal{S} = S^B$, where S represents the size of the search space per block and B represents the number of blocks. The size of the search space S per block can be calculated as $S = |\text{ConvOp}| \cdot |\text{KernelSize}| \cdot |F_i| \cdot |N_i| \cdot |\text{SkipOp}|$, where$| \cdot |$ represents the number of choices for each element. The sizes $\mathcal{S}_A, \mathcal{S}_B, \mathcal{S}_C$, and $\mathcal{S}_D$ of the search spaces A/B/C/D were calculated as $\mathcal{S}_A \approx 10^8, \mathcal{S}_B \approx 10^{14}, \mathcal{S}_C \approx 10^{15}$, and $\mathcal{S}_D \approx 10^{20}$ using the above formula. The size of the search space B/C/D is very large compared with search space A, which has a fixed number of filters. The obtained model may not be the optimal model architecture.

1) Search Space: To reduce the size of the search space, we introduce the relative filter size, which can be defined as $f \times F$, where f and F denote the ratio of change in the *number of output filters* and the number of reference filters, respectively. Although MnasNet used $\{16, 24, 32, 64, 96, 160, 320\}$ as reference filter size $\{F_i \in \boldsymbol{F} | i \in \mathbb{N}, i \leq 7\}$ based on the MobileNetV2 [26], we used $\{76, 88, 100, 112\}$ as reference filter size $\{F_i \in \boldsymbol{F} | i \in \mathbb{N}, i \leq 4\}$ based on the PyramidNet 18 [21] because of the number of layers. In this search space E, we adopt the above-mentioned relative ratio f rather than *the number of filters* as a search factor. The search space for the ith block is as follows.

1) *Search Space E:*
   a) *The ratio of change in the number of output filters* f_i: 0.75, 1.0, 1.25.
   b) *Other Parameters:* Same as search space C.

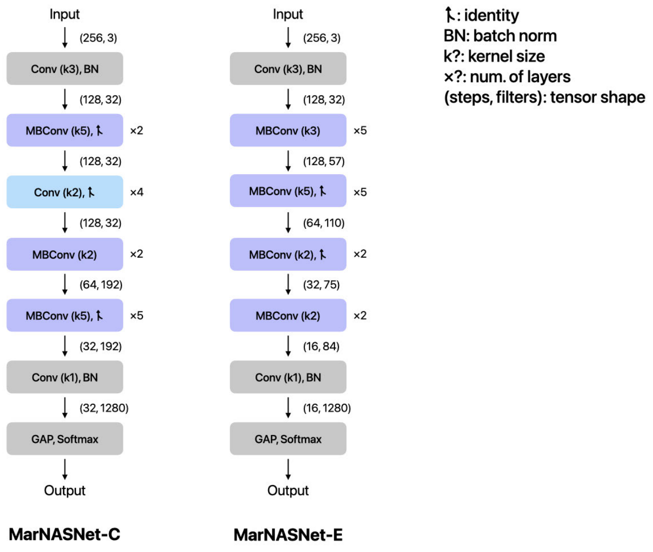


Fig. 9. MarNASNet-C and MarNASNet-E model overview.

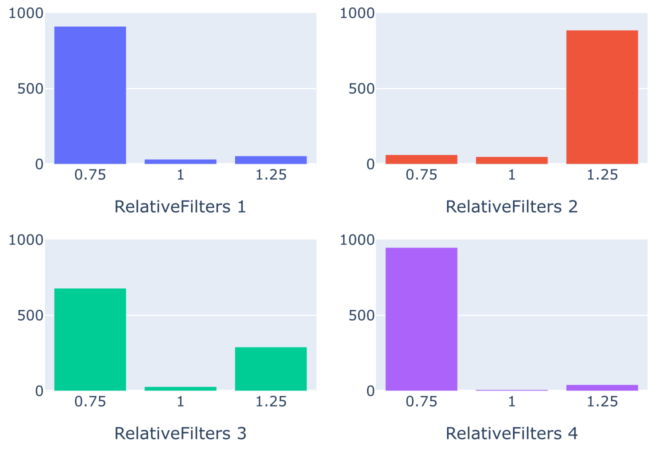


Fig. 10. Number of times each alternative is selected for each rate of change in the *number of output filters* f_i in search space D.

SERatio is the same as search space C. The size of the search space is $\mathcal{S}_E \approx 10^{10}$, which is smaller than search space B/C/D and close to the size of search space A.

2) MarNASNet-E: Fig. 9 shows model overview diagram of MarNASNet-E and MarNASNet-C, the models obtained by search space E. In search space E, $\{0.75, 1.25, 0.75, 0.75\}$ is selected as relative ratio $\{f_i \in \boldsymbol{f} | i \in \mathbb{N}, i \leq 4\}$. MBConv is selected for all *ConvOp*. Fig. 10 shows a bar chart of the number of times each choice of output filter change rate f_i is selected for each of the four blocks in search space E. All three options are selected in every block even though 1.0 is selected less frequently.

C. Experiments

Experiments to evaluate the performance of MarNASNet-E will be conducted similarly as in Section IV-D, with experiments to evaluate the accuracy and on-device performance. Table IV shows accuracy, model size, latency, and overall

TABLE IV
PERFORMANCE OF MARNASNET-C AND MARNASNET-E

Model	Accuracy [%] ⇑			Size	Latency	CPU
	HASC	UCI	WISDM	[MB] ⇓	[ms] ⇓	[%] ⇓
MarNASNet-C	**92.60**	**94.52**	88.92	3.08	**2.22**	**1.83**
MarNASNet-E	91.87	92.28	**89.73**	**1.25**	2.25	1.86

CPU usage on the three datasets MarNASNet-C and E. The better results of the two MarNASNets are highlighted in bold. The accuracy of MarNASNet-E is lower than that of MarNASNet-C on HASC and UCI but higher than that of MarNASNet-C on WISDM. The model size of MarNASNet-E is smaller than that of MarNASNet-C. This is because the maximum filter size in PyramidNet18, the reference model, is 112, and 0.75 is often selected as the output filter size. The latency and CPU usage were similar to those of MarNASNet-C. In conclusion, MarNASNet-E is not the best model architecture in terms of accuracy, but it is a good model considering the model size and other factors.

VI. CONCLUSION

While DL has been actively studied in HAR, the optimal model architecture that takes into account the on-device performance has not been clarified. In our previous work [42], we comprehensively evaluated the effectiveness of various CNN architectures proposed in the image recognition field without considering the model size and on-device latency. In this study, based on the evaluation of CNN models for image recognition, we explored MarNASNets, which are highly accurate and lightweight CNN models for sensor-based HAR, using Bayesian optimization of NAS. In addition, we developed Activitybench, an iOS app for evaluating the HAR models on an actual device. In the search and evaluation experiments of MarNASNets, we explored the model architectures of MarNASNet-A to -D by Bayesian optimization using multiple search spaces and evaluated the performance of the obtained models in terms of estimation accuracy, on-device latency, and on-device CPU usage. The performance of the existing CNN models was also evaluated in the same way and compared with MarNASNets.

MarNASNets explored achieved accuracy comparable to existing CNN models with smaller model sizes. While PyramidNet 18, a similar model size to ours, achieved accuracies of 91.48%, 92.56%, and 85.23% for HASC, UCI, and WISDM, respectively, MarNASNet-C achieved those of 92.60%, 94.52%, and 88.92%. Especially for HASC and UCI, MarNASNet-C achieved the highest accuracies despite the small model size. Their latency was also comparable to that of existing CNN models, enabling real-time on-device inference.

We describe the limitations of this study. In this study, we investigated the optimal model architecture based on the search space mentioned in Section III-A due to the time constraints of the search by NAS. Therefore, it is possible that a more efficient model architecture can be found using an expanded search space, such as inception module and transformer block by taking more time. We investigated the architecture of MarNASNets based on the validation accuracy of the HASC dataset. Our experiments demonstrated that MarNASNets performed effectively, not only on the HASC dataset, but also when alternative datasets were used for validation. However, there is potential to create a more accurate and computationally efficient model by incorporating different datasets and reward settings. In addition, the performance of MarNASNets may decrease for more complex HAR tasks, which we have not yet evaluated. As such, it would be prudent to develop a specialized approach for these complex HAR tasks in future work. More information on evaluating performances on other platforms or other devices, such as TensorFlow lite or Android smartphones, would help us show our method's robustness. In addition, incorporating on-device-performance metrics, such as model size and latency, into object functions can improve the on-device performance.

REFERENCES

[1] A. Cooper, A. Page, K. Fox, and J. Misson, "Physical activity patterns in normal, overweight and obese individuals using minute-by-minute accelerometry," *Eur. J. Clin. Nutrition*, vol. 54, no. 12, pp. 887–894, Dec. 2000.

[2] U. Ekelund, S. Brage, S. J. Griffin, and N. J. Wareham, "Objectively measured moderate- and vigorous-intensity physical activity but not sedentary time predicts insulin resistance in high-risk individuals," *Diabetes Care*, vol. 32, no. 6, pp. 1081–1086, Jun. 2009.

[3] J. Chen, K. Kwong, D. Chang, J. Luk, and R. Bajcsy, "Wearable sensors for reliable fall detection," in *Proc. IEEE Eng. Med. Biol. 27th Annu. Conf.*, Shanghai, China, 2005, pp. 3551–3554.

[4] K. Aizawa, "Digitizing personal experiences: Capture and retrieval of life log," in *Proc. 11th Int. Multimedia Model. Conf.*, Melbourne, VIC, Australia, 2005, pp. 10–15.

[5] A. Kobayashi et al., "Shaka: User movement estimation considering reliability, power saving, and latency using mobile phone," *IEICE Trans. Inf. Syst.*, vols. E94–D, no. 6, pp. 1153–1163, 2011.

[6] L. Bao and S. S. Intille SS, "Activity recognition from user-annotated acceleration data," in *Proc. Int. Conf. Pervasive Comput.*, 2004, Art. no. 1201317.

[7] Z. Chen, Q. Zhu, Y. C. Soh, and L. Zhang, "Robust human activity recognition using smartphone sensors via CT-PCA and online SVM," *IEEE Trans. Ind. Informat.*, vol. 13, no. 6, pp. 3070–3080, Dec. 2017.

[8] J. Yang, M. N. Nguyen, P. P. San, X. L. Li, and S. Krishnaswamy, "Deep convolutional neural networks on multichannel time series for human activity recognition," in *Proc. 24th Int. Joint Conf. Artif. Intell.*, 2015, pp. 1–7.

[9] F. Li, K. Shirahama, M. A. Nisar, L. Köping, and M. Grzegorzek, "Comparison of feature learning methods for human activity recognition using wearable sensors," *Sensors*, vol. 18, no. 679, pp. 1–22, Feb. 2019.

[10] J. Long, W. Sun, Z. Yang, and O. I. Raymond, "Asymmetric residual neural network for accurate human activity recognition," *Information*, vol. 10, no. 6, pp. 1–19, Jun. 2019.

[11] M. M. Hossain Shuvo, N. Ahmed, K. Nouduri, and K. Palaniappan, "A hybrid approach for human activity recognition with support vector machine and 1D convolutional neural network," in *Proc. IEEE Appl. Imag. Pattern Recognit. Workshop (AIPR)*, Oct. 2020, pp. 1–5.

[12] M. Zeng et al., "Convolutional neural networks for human activity recognition using mobile sensors," in *Proc. 6th Int. Conf. Mobile Comput., Appl. Services*, Nov. 2014, pp. 197–205.

[13] C. Xu, D. Chai, J. He, X. Zhang, and S. Duan, "InnoHAR: A deep neural network for complex human activity recognition," *IEEE Access*, vol. 7, pp. 9893–9902, 2019.

[14] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–14.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[16] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.

[17] N. Kawaguchi et al., "HASC challenge: Gathering large scale human activity corpus for the real-world activity understandings," in *Proc. 2nd Augmented Human Int. Conf.*, Mar. 2011, pp. 1–5.

[18] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.
[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst.*, vol. 1, 2012, pp. 1097–1105.
[20] C. Szegedy et al., "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.
[21] D. Han, J. Kim, and J. Kim, "Deep pyramidal residual networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6307–6315.
[22] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1800–1807.
[23] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2261–2269.
[24] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7132–7141.
[25] A. G. Howard et al., "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.
[26] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.
[27] A. Howard et al., "Searching for MobileNetV3," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1314–1324.
[28] 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.
[29] M. Tan et al., "MnasNet: Platform-aware neural architecture search for mobile," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2815–2823.
[30] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.
[31] Y. Chen and Y. Xue, "A deep learning approach to human activity recognition based on single accelerometer," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 2015, pp. 1488–1492.
[32] C. Han, L. Zhang, Y. Tang, W. Huang, F. Min, and J. He, "Human activity recognition using wearable sensors by heterogeneous convolutional neural networks," *Exp. Syst. Appl.*, vol. 198, no. 15, 2022, Art. no. 116764.
[33] W. Huang, L. Zhang, H. Wu, F. Min, and A. Song, "Channel-equalization-HAR: A light-weight convolutional neural network for wearable sensor based human activity recognition," *IEEE Trans. Mobile Comput.*, early access, May 13, 2022, doi: 10.1109/TMC.2022.3174816.
[34] F. Ordóñez and D. Roggen, "Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition," *Sensors*, vol. 16, no. 1, p. 115, Jan. 2016.
[35] K. Xia, J. Huang, and H. Wang, "LSTM-CNN architecture for human activity recognition," *IEEE Access*, vol. 8, pp. 56855–56866, 2020.
[36] N. Dua, S. N. Singh, and V. B. Semwal, "Multi-input CNN-GRU based human activity recognition using wearable sensors," *Computing*, vol. 103, no. 7, pp. 1461–1478, Jul. 2021.
[37] N. T. Hoai Thu and D. S. Han, "HiHAR: A hierarchical hybrid deep learning architecture for wearable sensor-based human activity recognition," *IEEE Access*, vol. 9, pp. 145271–145281, 2021.
[38] S. Mekruksavanich, N. Hnoohom, and A. Jitpattanakul, "A hybrid deep residual network for efficient transitional activity recognition based on wearable sensors," *Appl. Sci.*, vol. 12, no. 10, pp. 1–21, 2022.
[39] D. Thakur, S. Biswas, E. S. L. Ho, and S. Chattopadhyay, "ConvAE-LSTM: Convolutional autoencoder long short-term memory network for smartphone-based human activity recognition," *IEEE Access*, vol. 10, pp. 4137–4156, 2022.
[40] S. Jameer and H. Syed, "Deep SE-BiLSTM with IFPOA fine-tuning for human activity recognition using mobile and wearable sensors," *Sensors*, vol. 23, no. 9, p. 4319, Apr. 2023.
[41] D. Bhattacharya, D. Sharma, W. Kim, M. F. Ijaz, and P. K. Singh, "Ensem-HAR: An ensemble deep learning model for smartphone sensor-based human activity recognition for measurement of elderly health monitoring," *Biosensors*, vol. 12, no. 6, p. 393, Jun. 2022.
[42] Z. Zhongkai, S. Kobayashi, K. Kondo, T. Hasegawa, and M. Koshino, "A comparative study: Toward an effective convolutional neural network architecture for sensor-based human activity recognition," *IEEE Access*, vol. 10, pp. 20547–20558, 2022.
[43] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Oritz, "A public domain dataset for human activity recognition using smartphones," in *Proc. 21st Int. Eur. Symp. Artif. Neural Netw., Comput. Intell. Mach. Learn.*, 2013, pp. 437–442.
[44] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *ACM SIGKDD Explor. Newsl*, vol. 12, pp. 78–82, Mar. 2010.
[45] Y. Tang, Q. Teng, L. Zhang, F. Min, and J. He, "Layer-wise training convolutional neural networks with smaller filters for human activity recognition using wearable sensors," *IEEE Sensors J.*, vol. 21, no. 1, pp. 581–592, Jan. 2021.

Satoshi Kobayashi received the B.S. degree in engineering from the University of Fukui, Fukui, Japan, in 2020, and the M.S. degree from the Graduate School of Engineering, University of Fukui, in 2022.

He is currently an Engineer with Keywalker, Inc., Tokyo, Japan. His research interests include human activity recognition and deep learning application.

Tatsuhito Hasegawa (Member, IEEE) received the Ph.D. degree in engineering from Kanazawa University, Kanazawa, Japan, in 2015.

From 2011 to 2013, he was a System Engineer with Fujitsu Hokuriku Systems Ltd., Kanazawa, Japan. From 2014 to 2017, he was an Assistant with Tokyo Healthcare University, Tokyo, Japan. From 2017 to 2020, he was a Senior Lecturer with the Graduate School of Engineering, University of Fukui, Fukui, Japan. He is currently an Associate Professor with the University of Fukui. His research interests include human activity recognition (HAR), application of deep learning, and intelligent learning support systems.

Dr. Hasegawa was a member of Information Processing Society of Japan (IPSJ).

Takeru Miyoshi received his M.S. degree in engineering from Kanazawa University, Kanazawa, Japan, in 2013, where he is pursuing a Ph.D. degree in engineering.

He is currently an Assistant Professor with the National Institute of Technology, Ishikawa College, Tsubata, Japan. His research interests include computer vision on edge devices and human activity recognition (HAR), and applying deep learning.

Prof. Miyoshi is a member of Information Processing Society of Japan (IPSJ).

Makoto Koshino received his Ph.D. degree from Kanazawa University, Kanazawa, Japan, in 2004.

He is currently an Associate Professor with the National Institute of Technology, Ishikawa College, Tsubata, Japan. His current research interests include intelligent smartphones.

Dr. Koshino is a member of Institute of Electronics, Information and Communication Engineers (IEICE), Information Processing Society of Japan (IPSJ), and Japanese Society for Artificial Intelligence (JSAI).