Multitask Deep Learning for Human Activity, Speed, and Body Weight Estimation Using Commercial Smart Insoles

Jaeho Kim, Hyewon Kang, Jaewan Yang, Haneul Jung, Seulki Lee, *Member, IEEE*, and Junghye Lee

***Abstract*—Healthcare professionals and individual users use wearable devices equipped with various sensors for healthcare management. Recently, the joint usage of artificial intelligence and these wearable sensors has played an essential role in healthcare management by providing a wide range of applications, such as fitness tracking, gym activity monitoring, patient rehabilitation monitoring, and disease detection. These tasks eventually aim to enhance personal well-being and better manage the user's physical health by monitoring different activity types and body weight changes. Here, we present an efficient multitask learning (MTL) framework based on commercial smart insoles that can solve three tasks related to physical health management: 1) activity classification; 2) speed estimation; and 3) body weight estimation. Our multitask framework converts the sensor data from the smart insole to a recurrence plot, which shows significant performance improvement compared to processing the raw time-series data. In addition, we utilized a modified MobileNetV2 as our backbone network, which has a total parameter of less than 100K and a computational budget of 0.34G of multiply-accumulate operations. Furthermore, we collected a vast data set from 72 users carrying out 16 experiments, which contains the largest number of people for MTL purposes using smart insoles. Extensive experiments show that the proposed MTL framework is extremely efficient while outperforming or leading to comparable performance against single-task models.**

***Index Terms*—Body weight estimation (BWE), deep learning, human activity recognition (HAR), multitask learning (MTL), recurrence plot (RP), smart insole, speed estimation (SE).**

Manuscript received 21 February 2023; accepted 11 April 2023. Date of publication 14 April 2023; date of current version 7 September 2023. This work was supported in part by the Technology Development Program funded by the Ministry of SMEs and Startups (MSS), South Korea, under Grant S2946437; in part by the National Research Foundation of South Korea (NRF) Grant funded by the Korea Government (MSIT) under Grant 2020R1C1C1011063; in part by the Ministry of Education of the Republic of South Korea; and in part by the NRF of South Korea under Grant NRF-2020S1A3A2A02093277 and Grant NRF-2022R1I1A4069163. *(Jaeho Kim and Hyewon Kang are co-first authors.) (Corresponding author: Junghye Lee.)*

Jaeho Kim is with the Artificial Intelligence Graduate School, Ulsan National Institute of Science and Technology, Ulsan 44919, South Korea (e-mail: kjh3690@unist.ac.kr).

Hyewon Kang is with the Department of Industrial Engineering, Ulsan National Institute of Science and Technology, Ulsan 44919, South Korea (e-mail: hwkang91@unist.ac.kr).

Jaewan Yang and Haneul Jung are with the Research and Development Department, Gilon, Inc., Seongnam 13438, South Korea (e-mail: jayyang@gilon.co.kr; cielo@gilon.co.kr).

Seulki Lee is with the Department of Computer Science and Engineering and the Artificial Intelligence Graduate School, Ulsan National Institute of Science and Technology, Ulsan 44919, South Korea (e-mail: seulki.lee@unist.ac.kr).

Junghye Lee is with the Technology Management, Economics and Policy Program and the Graduate School of Engineering Practice, Seoul National University, Seoul 08826, South Korea (e-mail: junghye@snu.ac.kr).

Digital Object Identifier 10.1109/JIOT.2023.3267335

I. Introduction

ARTIFICIAL intelligence (AI) equipped with numerous sensors is becoming more and more in use in our everyday lives. The smart healthcare industry is one domain to which such device has significantly contributed. Fig. 1 shows the quadrant of sensor-based AI application cases in healthcare, consisting of two axes: 1) sensor type (wearable/nonwearable) and 2) demander (healthcare professionals/personal consumer) [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12]. In particular, the third and fourth quadrants of wearable device-related AI applications encompass practical use cases, including fitness tracking, monitoring gym activities, aiding in patient rehabilitation, and detecting diseases. The ultimate goal of these tasks can be summarized by obtaining physical well-being and health treatment, whose success depends on monitoring exercises and body weight changes [13], [14], [15]. In this article, we focus on human activity recognition (HAR) and body weight estimation (BWE), which are representative AI usages to achieve the goal of the third and fourth quadrants. Here, HAR refers to a comprehensive study of identifying any named actions or movements of human activity based on raw sensor signals, such as activity classification (AC) and speed estimation (SE) [16], [17]. Similarly, BWE is the task of approximating a person's body weight [18], [19], [20]. Thus, the majority of the tasks in the two quadrants can be achieved by combining HAR and BWE technologies, which enable simultaneous monitoring of body weight change and activity posture. However, so far, most research in developing AI models for HAR or BWE in a mobile environment focuses on an AI model that targets a single task, either HAR or BWE, but not both due to the tradeoffs between the number of functions and the overall computation cost that need to be considered in edge devices. Unfortunately, deploying multiple single-task models to mobile devices is a nontrivial problem due to the extra constraint of limited model capacity on mobile platforms. Besides, retraining or updating several single-task models over-the-air (OTA) makes model maintenance harder than a single multitask model [21], [22].

We focus our research on smart insoles, as they are effective data collection equipment that is low cost, low powered,

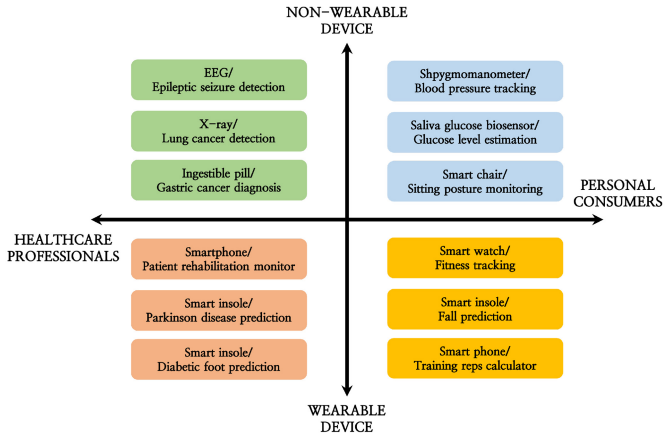

Fig. 1. Quadrant of AI application in the healthcare industry based on various sensor devices, including real-world examples.

and mobile. Since the smart insole receives plantar pressure and acceleration feedback directly from a person, it is advantageous for monitoring and measuring all activities that put a load on the body [23]. Despite the usefulness of smart insoles, much AI-based healthcare research using smart insoles has limited its use in the HAR study with only a handful number of BWE studies. Moreover, the HAR studies based on the smart insoles have focused on obtaining high predictive performance and have thus used hand-crafted smart insoles made in the lab [24], [25]. Multiple sensors are included for data collection purposes, but the increased amenities translate to higher costs and are less likely to be commercial. Increased communication cost and instability of wirelessly transporting data without data losses to mobile devices are additional concerns for an increased number of sensors. Thus, there is a need for research focused on solving HAR and BWE together based on commercial smart insoles, which generally have fewer numbers of embedded sensors.

In this article, we introduce a lightweight multitask learning (MTL) deep learning framework that addresses three interconnected tasks related to HAR and BWE using a commercial smart insole. These tasks include predicting the type of human activity, the user's speed, and the user's body weight. Our proposed network uses MobileNetV2 as its backbone model to process time-series input, which is first converted into a recurrence plot (RP). Our experiments demonstrate that using RP-based input can lead to outstanding performance compared to using raw time signals. The main contributions of our work are as follows.

1) We present a lightweight MTL deep learning model using commercially available smart insoles to deliver wearable-device-related AI healthcare services (i.e., the third and fourth quadrants), performing three crucial tasks: AC, SE, and BWE. To the best of our knowledge, this is the first time these key tasks have been combined into a unified framework. Our framework can be extended to other edge-device settings, such as smartphones and smartwatches, as it is designed with an MTL architecture for real-world applications.
2) Our MTL model is comparable to single-task models in terms of effectiveness and is three times more efficient than separate single-task models, making it a practical choice for mobile services. We achieved this by incorporating an uncertainty-weighted loss to balance the task losses, using a reduced version of the RP-based MobileNetV2 architecture as the backbone. The layers of the MobileNetV2 were strategically trimmed to strike a balance between effectiveness and efficiency.
3) We have created a novel data set[1] for multivariate time-series (MTSs) analysis, composed of raw sensor signals obtained from 72 individuals engaging in various activities using a commercial smart insole fitted with a 3-axis accelerometer and four force-sensing resistor (FSR) sensors per foot. The collection of labeled MTS data can be challenging as it requires human supervision during data collection, resulting in many open-source data sets having a small number of subjects, limited tasks, and channels. Our data set stands out for its diversity, featuring 72 subjects, the ability to address multiple problems in deep learning (seven class classification, two regression problems, and MTL) using 14 channels, and being multilabeled. This data set represents a valuable resource for researchers working on time-series applications in the future.
4) Our study also provides new insights into BWE, a crucial yet under-researched area in wearable-device-related AI healthcare services. By evaluating the largest number of subjects to date, we identified critical locations of the foot region and specific experimental conditions that help to improve BWE performance. These findings will substantially benefit the research community and serve as a valuable reference for future studies.

The remainder of this article is organized as follows. Section II reviews previous works of AC, SE, and BWE. We then describe our data collection protocol and the collected MTS data set for our task in Section III. Subsequently, we introduce and elaborate on our proposed methods in detail in Section IV. Sections V and VI entail all the details of our experimental setups and their results. In Section VII, we provide our discussions regarding BWE. Finally, we conclude our results in Section VIII.

II. Related Works

This section summarizes previous works with single-task and multitask models related to our three main tasks: 1) AC; 2) SE; and 3) BWE. It should be noted that not much research has been conducted for SE and BWE compared to AC. However, common to all three tasks, early works focused on using conventional machine learning algorithms, such as support vector machine (SVM), k-nearest neighbors (KNNs), and decision trees (DTs) [26], [27], [28], [29], [30]. Conventional machine learning methods are easy to implement and have proven to be mathematically sound. However, their performances usually depend on knowledge-driven feature engineering of the input, such as the generation of statistically transformed features (e.g., mean and median), and feature selection and extraction, which requires in-depth domain

[1] https://github.com/Gilon-Inc/GILON-MULTITASK-DATASET

expertise. Compared to conventional machine learning models, deep learning is known to automatically extract meaningful features from the input through multiple hidden layers and show better model performances. Accordingly, deep learning methods, such as recurrent neural networks (RNNs) and convolutional neural networks (CNNs) have been the mainstream in recent works [20], [25], [31], [32], [33], [34], [35].

A. Activity Classification

Conventional machine learning-based AC heavily relied on numerous feature engineering techniques for high predictive performance. For example, Atallah et al. [26] developed an activity of daily living classifier based on the features generated from the fast Fourier transform (FFT). Similarly, Yang et al. [28] used the first-order inertial filtering and Kalman filtering methods to reduce the high-frequency noise from the raw time signals and utilized a KNN classifier to classify different types of lower limb motion. Merry et al. [27] applied SVM, DT, and KNN to classify workplace behaviors, such as sitting, standing, and walking based on hand-crafted features, such as mean, mode, median, and sum from univariate time-series data. The features were then preselected before model training based on various filter-based feature selection methods, such as chi-square, Fisher score feature, Gini index, and info-gain. In addition to the complicated feature engineering process, this study utilized 48 plantar FSR sensors for right and left insoles, which makes commercialization challenging. In general, we notice that there is no one-size-fits-all strategy for generating features; rather, the strategy is very task-dependent. In addition, we contend that the approximate 10 subjects utilized in the aforementioned studies is insufficient for generalization performance.

Deep learning-based AC works can be categorized into two, based on how the input data is processed for model training. The first approach applies an RNN or 1-D CNN model to the raw input times series data. The suggested method of Agarwal and Alam [31] employed a simple two-layer LSTM to classify six activities based on smartphone sensors. Ronao and Cho [32] validated the performance of a regularized 1-D CNN model in conjunction with other traditional machine learning models, such as SVM on the HAR data set collected from smartphone sensors. However, the performance gain from the deep network compared to other machine learning baseline models was minute. The second approach converts time-series signals to image-based data, thus making better use of models originating to process image data. Lu and Tong [33] converted raw time-series data to an RP and classified different activities using a modified ResNet model. This improved RF, SVM, and LSTM models with the raw time-series data, but it needs to be further studied in a larger data set.

B. Speed Estimation

The works of SE can be grouped into two: classification models classifying the broad range of speed and regression models regressing the exact speed value. Mannini and Sabatini [29] suggested an SVM-based speed classification model through a data set collected from 30 subjects who wore a single thigh-mounted triaxial accelerometer. The model aimed to classify speed-related tasks, including running and walking. This work is an early stage of SE studies, which can predict only the range of speed. As for the deep learning approach for SE classification, Low et al. [34] proposed a bidirectional LSTM speed classifier that classifies three walking types: slow, standard, and fast.

In contrast to the rough classification of speed, Wei et al. [30] proposed an artificial neural network-based SE model, which predicted the precise speed value with an average root-mean-square error (RMSE) of 0.003 ± 0.043 m/s. However, the result was obtained from four subjects, which is insufficient to guarantee the model's reliability. Seethi and Bharti [35] used a 1-D CNN for SE regression. The convolution operation is performed on accelerometer and gyroscope sensors separately, and the output hidden features are concatenated before being processed by an output layer. The data was collected from wrist-worn wearable sensors, and 15 adults were involved in this study. In general, previous works on SE train and validate their work with less than 30 subjects, which lowers the generalization performance.

C. Body Weight Estimation

Sazonova et al. [18] proposed a simple formula-based equation to predict the body weight based on pressure sensors in a smart insole. The body weight was estimated when the subject was in a standing posture, resulting in a rough estimate of 10.52 kg (RMSE). The equation has been tested on nine subjects. Kim and Hong [20] carried out the first deep learning-based BWE study, where the team utilized a simple deep neural network to measure body weight for those with a physical disability. A smart mat was embedded with 128 FSR sensors where the subjects were asked to lie down for measurement.

D. Multitask-Based Works

In addition to the single-task-based models, multitask-based models for different HAR tasks have been suggested. Under the assumption that the MTL model shows comparable performance and the number of parameters is similar to single-task models, it would be beneficial to deploy an MTL model for all parties involved. Barut et al. [36] proposed an LSTM-based multitask model performing AC and activity intensity estimation of each activity. A measurement device was attached to the waist for data collection, and the model was verified on ten subjects, with additional evaluation on a public data set. Martindale et al. [25] used a combination of CNN and RNN models to predict the type of activity and gait cycle based on inertial measurement unit sensors.

Generally, MTL models show improved performance compared to single-task models when the tasks are related and have beneficial knowledge shared between tasks. However, due to the difficulty in training an efficient MTL model, there seems to be a lack of research on MTL models that is suitable for tasks in the third and fourth quadrants of sensor-based AI applications.

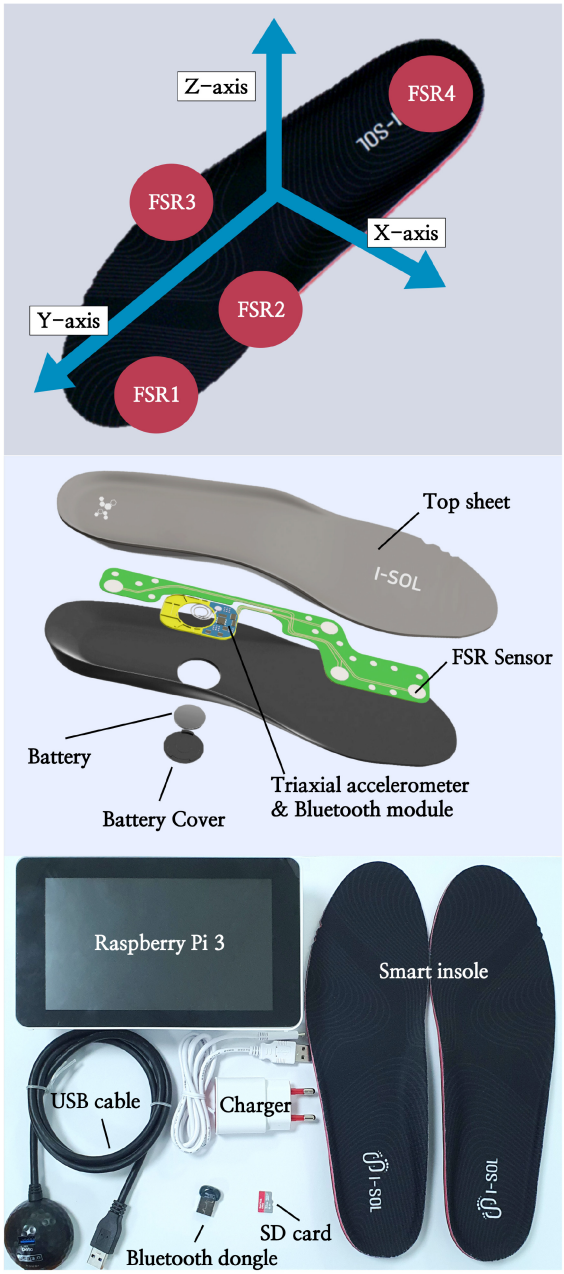

Fig. 2. (Top) Sensor location in the smart insole. (Middle) Product drawing of the smart insole used for data collection. (Bottom) The data collection tools used.

III. Data Collection

A. Collection Tools

In this work, we used a commercial smart insole made by Gilon, INC. for data collection. The smart insole comes in various sizes with 5-mm units, ranging from 230 to 280 mm. All participants were able to select the size appropriate to their feet. As shown in Fig. 2, a single insole has a 3-axis accelerometer sensor and four FSR sensors embedded, making it a total of 14 different sensors for a pair of insoles. The accelerometer sensors measure the acceleration along the x-, y-, and z-axis, and the FSR sensors detect the relative change in force at four different key points. The data were collected at 40 Hz using the Raspberry Pi 3 system.

TABLE I
Experiment Description and Labels Assigned

Experiment	Task	Activity	Speed *(km/hr)*	Body Weight *(kg)*
EXP 01	Stand still on Ground	Stand Still	0	*N.U*
EXP 02	Stand still on Ground with Bag			
EXP 03	Treadmill 3km/hr	Treadmill Walk	3	
EXP 04	Treadmill 3.5km/hr		3.5	
EXP 05	Treadmill 4km/hr		4	
EXP 06	Treadmill 4.5km/hr		4.5	
EXP 07	Treadmill 5km/hr		5	
EXP 08	Treadmill 5.5km/hr		5.5	
EXP 09	Treadmill 7km/hr	Treadmill Run	7	O
EXP 10	Treadmill 7.5km/hr		7.5	O
EXP 11	Treadmill 8km/hr		8	O
EXP 12	Walk on Ground	Ground Walk	*N.U*	*N.U*
EXP 13	Walk on Ground with Bag			
EXP 14	Squat	Squat		
EXP 15	Lunge	Lunge		
EXP 16	Jumping Jack	Jumping Jack		

N.U: Not Used

B. Experiment Design

A total of 72 subjects (38 males and 34 females) participated in our experiments, with ages ranging from 19 to 65 (35±11.7). All subjects engaged in 16 different tasks, and the details of each task can be found in Table I. The data were collected for 4–50 s for each task, and participants were allowed to take a 1-min break at the end of each task. To ensure that the data collection and experimental outcomes are not affected by the sequence in which the actions are taken, half of the participants carried out the experiment in the order of experiments 1–16, while the other half did it in reverse order. The study was approved by the Ulsan National Institute of Science and Technology Institutional Review Board (IRB).

The experiment was designed to effectively and accurately collect the labels for our MTL framework. In detail, each subtasks required participants to perform one specific action on the ground floor or treadmill. For HAR, actions, such as standing still, walking on the ground, squatting, lunging, and jumping jacks were performed on the floor, and treadmill walking and running were done on the treadmill. We used all experiment sets for the task of AC. Subsequently, the label for SE was acquired while participants were staying still or carrying out treadmill walking and running, thus utilizing

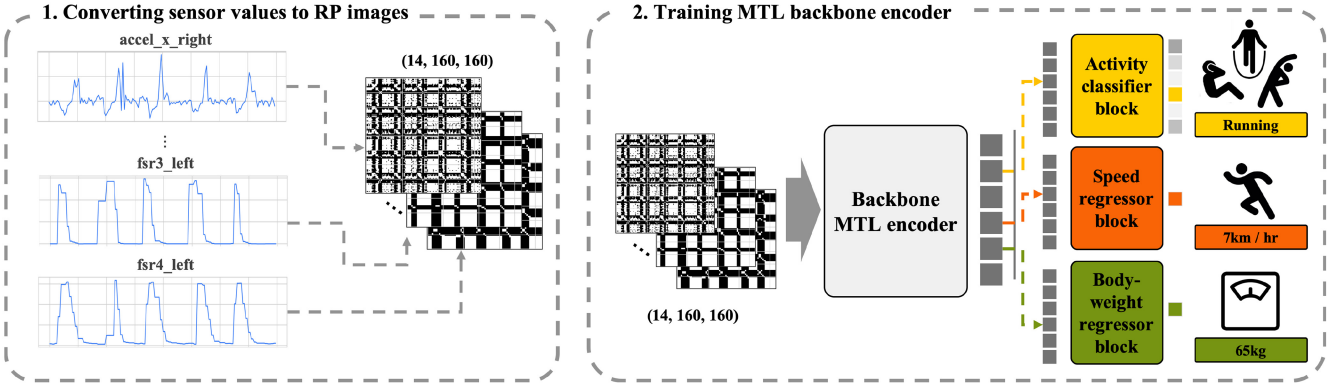

Fig. 3. Overview of our MTL framework. First, the raw sensor signals collected from the smart insole are converted to RP images. Second, the RP images are fed into the MTL encoder, which simultaneously predicts the result of AC, SE, and BWE.

experiment sets EXP 01 to EXP 11. The treadmill was set to the target speed, and participants were asked to either walk or run based on the target speed. Speed labels were acquired for 0, 3, 3.5, 4, 4.5, 5, 5.5, 7, 7.5, and 8 km/h. The range between 5.5 and 7 km/h was excluded in our experiment design as participants either jogged or walked fast, making the activity type ambiguous. Finally, we utilized experiment sets EXP 09 to EXP 11 for BWE. For EXP 02 and 13, we asked participants to wear a backpack that contained books of weight 3.4 kg, but these experiment subsets were later found to be not helpful and were not used in BWE. We discuss the reasonings behind this selection of subsets in Section VII. The users' body weight was measured twice before the experiments, and the average was used as the final body weight label for individual users.

IV. Methodology

A. MTL Framework

Here, we illustrate the general overview of our MTL framework, which performs three related but different tasks: AC, SE, and BWE of the smart insole user. The overall flow of our work is presented in Fig. 3. First, we segment the raw sensor data into time-series samples. As a data preprocessing step, we apply 4 s successive time windows with a 3-s overlapping window [37]. Since the data is collected from 14 sensors at 40 Hz, the data for 4 s consists of 160 signal sequences with 14 features. Here, we consider a specific case in which we process a raw time series of 2000 time steps (40 Hz $\times$ 50 s) collected from a single user conducting one experiment. With our preprocessing strategy, we obtain an approximate of $([(2000-160)/(160-120)]+1)=47$ samples for this experiment. Each raw time-series sample has the shape of (14, 160). Second, since our sample has 14 features, 14 different RPs are generated by converting each feature into an RP. These RPs are stacked into channels, resulting in an image-like data set of shape (14, 160, 160), which is processed by the shared backbone module. The shared representation output from the backbone module is then fed as an input to three task-specific, fully connected neural networks: 1) activity classifier; 2) speed regressor; and 3) body weight regressor. The activity classifier outputs a class probability of the actions while the speed regressor and body weight regressor return a single estimate of the speed and body weight.

B. Notations

For all time-series sample $i=1$ to N, let $\mathbf{X}_i \in \mathbb{R}^{d\times T}$ be an MTS data of the ith sample where d is the number of features with T time steps of view. Here, $\mathbf{X}=[\mathbf{x}_1,\ldots,\mathbf{x}_d]$ is a collection of d number of univariate time series $\mathbf{x}_j=[x_1,x_2,\ldots,x_T]^T$, where $j\in\{1,\ldots,d\}$. Given N number of samples, $\{\mathbf{X}_i\}_{i=1}^N$, the MTL model $F_{\boldsymbol{\theta}}$ processes the ith MTS sample $\mathbf{X}_i$ to output $\mathbf{y}_i^{\text{activity}}\in\mathbb{R}^7$, $y_i^{\text{speed}}\in\mathbb{R}$, and $y_i^{\text{bodyweight}}\in\mathbb{R}$.

C. Recurrence Plot

The RP was developed by Eckmann et al. [38], which is one of the image encoding techniques for time-series data. The image visualizes the recurrence behavior between time points, such as periodicity or irregular cyclicity, a typical pattern observable in a nonlinear dynamical system. Recently, the RP has been widely used in deep learning to transform a univariate time-series data into a 2-D image for CNN. It has been shown that the RP brings performance improvement compared to using the raw time-series data in several applications [39], [40].

To transform the univariate time-series data into the 2-D image (RP), the time delay τ and embedding dimension m hyperparameters must be set. Specifically, from a univariate time series $\mathbf{x}$, we can generate $L\triangleq T-(m-1)\tau$ number of new vectors $\mathbf{s}_1,\ldots,\mathbf{s}_L$, where $\mathbf{s}_k$ is a vector defined as in

$$\mathbf{s}_k=\left[x_k,x_{k+\tau},x_{k+2\tau},\ldots,x_{k+(m-1)\tau}\right]^T. \tag{1}$$

In detail, the RP image is constructed by calculating the pairwise distance between all $\mathbf{s}_{k=1,\ldots,L}$ which results in an $L\times L$ sized image. Specifically, each elements of RP can be defined as follows:

$$\text{RP}_{(k,l)}=\mathbb{H}(\varepsilon-||\mathbf{s}_k-\mathbf{s}_l||)\ \forall k,l\in\{1,\ldots,L\}. \tag{2}$$

Here, ε is a threshold distance, $\mathbb{H}$ is a heaviside step function, and $||.||$ is the norm function. $\text{RP}_{(k,l)}=1$ when the norm between $\mathbf{s}_k$ and $\mathbf{s}_l$ is smaller than ε, and $\text{RP}_{(k,l)}=0$ otherwise.

D. MobileNetV2

The MobileNetV2 architecture [41] is used as the shared backbone for our experiment. This model is known for its inverted residual blocks, and linear bottlenecks, which improve its performance, and its lightweight and efficient design makes

TABLE II
CUSTOMIZED MOBILENETV2

Input	Operator	t	c	n	s	p
$160^2 \times 14$	conv2d 1×1	-	32	1	1	1
$162^2 \times 32$	bottleneck	1	16	1	1	1
$162^2 \times 16$	bottleneck	6	24	2	2	1
$81^2 \times 24$	bottleneck	6	32	3	2	1
$41^2 \times 32$	bottleneck	6	64	1	1	1
$41^2 \times 64$	conv2d 1×1	-	128	1	1	0
$41^2 \times 128$	adaptive avgpool 1×1	-	-	1	-	-
$1^2 \times 128$	conv2d 1×1	-	64	1	1	0
AC	Linear(64, 7)	-	-	-	-	-
SE	Linear(64, 1)	-	-	-	-	-
BWE	Linear(64, 1)	-	-	-	-	-

The architecture of each operator in our model is based on that of the original MobileNetV2 paper [41]. To provide clarity, the notation used in the paper includes the following symbols: t represents the expansion factor, c represents the output channel, n represents the number of repetitions, s represents the stride, and p represents the padding size. These symbols are used to indicate the design choices for each operator in the model.

it suitable for mobile applications. To adapt it to our specific use case, we made slight modifications to increase parameter efficiency while preserving its original design principles.

We modified the original seven-bottleneck MobileNetV2 architecture by using only the first three and last bottlenecks and added a 1×1 convolution at the front and after the MobileNetV2, as well as a global averaging layer. We have provided a detailed architecture structure in Table II. The 1×1 convolution added at the very beginning and end of the MobileNetV2 and the global averaging layer, helped us to adjust the number of channels and aggregate the activations in a 1-D vector of size 128, which allowed us to achieve a balance between good performance and reduced computational complexity and memory allocation requirements. We observed that the number of parameters increased exponentially in the later bottlenecks of the original MobileNetV2, but the performance of our tasks did not necessarily increase with the increased layers. Thus, we removed them to achieve a better balance of performance and efficiency. A detailed ablation on the number of layers in the MobileNetV2 and its corresponding performance is shown in Appendix A.

E. Multitask Loss

In our proposed MTL approach, we defined a loss function that is composed of three different terms. The first term $\mathcal{L}_1$, is a cross-entropy (CE) loss for AC. The second term $\mathcal{L}_2$, and the third term, $\mathcal{L}_3$, are mean-squared error (MSE) losses for SE and BWE, respectively. We chose CE and MSE losses as they are simple and widely used for classification and regression tasks, respectively [42], [43], [44], [45]

$$\mathcal{L}_1 = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{ic}\cdot\log\hat{y}_{ic} \tag{3}$$

$$\mathcal{L}_2, \mathcal{L}_3 = \frac{1}{N}\sum_{i=1}^{N}||y_i-\hat{y}_i||_2^2 \tag{4}$$

$$y_{ic}\in\{0,1\}, \hat{y}_{ic}\in[0,1], y_i, \hat{y}_{ic}\in\mathbb{R}. \tag{5}$$

Here, y and $\hat{y}$ represents the true and predicted value, respectively. N is the total number of samples and C represents the number of total classes. By jointly minimizing these loss terms, MTL training becomes robust to overfitting as each task receives information from another task-specific learning through the shared backbone network, improving performance relative to single-task training. However, training a multitask model is challenging as it requires a delicate balance between tasks-specific losses [46]. Simply averaging all the taskwise losses may degrade the multitask model's overall performance since it might cause negative transfer between tasks due to the different scales of the losses across tasks and the complexity of each task. For example, the scale in which the CE loss works differs from that of a MSE loss. Moreover, some tasks may be more difficult to train compared to other tasks. Thus, the loss balancing coefficients should be well measured and calibrated. One naive solution is to perform a grid search between the coefficients and use the combination which results in the lowest error. Unfortunately, such a naive grid search is impractical in most cases as the search space grows exponentially with the number of tasks. Besides, a fixed coefficient cannot respond to the dynamics of each loss in training. With the above consideration, we utilized the uncertainty-based weighting method from the work of [47].

Overall, the multitask loss function based on the uncertainty-based method maximizes the data's Gaussian likelihood by considering each task's homoscedastic uncertainty. The homoscedastic uncertainty σ is a learnable parameter with the final loss function balanced as

$$\mathcal{L}_{\text{total}} = \frac{1}{\sigma_1^2}\mathcal{L}_1 + \frac{1}{2\sigma_2^2}\mathcal{L}_2 + \frac{1}{2\sigma_3^2}\mathcal{L}_3 + \log\sigma_1 + \log\sigma_2 + \log\sigma_3. \tag{6}$$

As the uncertainty parameter works as the loss balancing coefficients throughout the training process, the losses between tasks are better adjusted than a uniform loss. Here, adding the logarithms of uncertainty parameters works as a regularizer to penalize when the uncertainties become too large.

V. EXPERIMENT

Here, we highlight the objectives of our experiments. We first compared the effectiveness and efficiency of our MTL models against single-task models. Subsequently, we carried out experiments to determine the optimal backbone model and multitask loss for our MTL framework. Finally, we look into the implementation details, including the evaluation scenario and the hyperparameter used for each experiment.

A. Experiment Overview

1) MTL Versus Single-Task: We conducted a performance comparison of our MobileNetV2-based MTL model to single-task MobileNetV2 models on a single V100 GPU with an Intel Xeon Gold 6254 CPU. All models were compiled using PyTorch 1.8.2 [48]. The model performance was compared in all AC, SE, and BWE tasks. To evaluate the efficiency of the model in a resource-constrained environment, we used Nvidia Jetson Nano with 4 GB of memory to measure the inference speed. The Jetson Nano is a popular edge device for

deep learning performance evaluation and is equipped with a 128-core NVIDIA Maxwell GPU and Cortex A57 CPU. To precisely measure the inference speed, we synchronized the CPU thread and CUDA stream and set the batch size to 1 with a GPU warmup step of 10 instances in advance.

2) Backbone Model Selection: We show the MobileNetV2 model performance compared to other deep learning models, such as LSTM [49] and GRU [50]. The LSTM and GRU were chosen as baselines as they are commonly used in other works for handling raw time-series data [51], [52], [53]. For the LSTM and GRU models, we used two hidden layers with a hidden unit size of 256, and applied a dropout rate of 0.3 to reduce overfitting. The hidden output representations from all the layers were then passed through max pooling and average pooling, and concatenated for the final fully connected layer. This resulted in 871K and 670K parameters for LSTM and GRU models, respectively, in the single-task models.

3) Comparison Between MTL Loss Functions: Multitask models exploit knowledge from multiple tasks, often showing improved performance compared to single-task models. However, incorporating irrelevant task knowledge or being unable to balance the multiple losses in an effective manner can lead to degraded performance, known as "negative transfer." To mitigate this, it is crucial to select an optimal loss function for each task. Here, we compared three different loss functions: 1) uniform loss; 2) random loss weighting (RLW); and 3) uncertainty-weighted loss. The uniform loss function applies a uniform weighting between the loss incurred in each task. RLW uses randomly sampled weights from a normal distribution and has been shown to result in stable training. The original paper argues that RLW should be used as a litmus test for MTL problems. The uncertainty-weighted loss function assumes that the noise in each observed data point follows a Gaussian distribution, allowing the model to learn the uncertainty per task and optimize the weighting coefficients for each task's loss.

B. Implementation

1) Deployment Scenario Under Mobile Conditions: In this section, we outline the deployment scenario of our MTL model in a mobile application. We distinguish the training and inference phase. Our scenario involves training and optimizing the model on a server and deploying the trained model weights to the mobile device for offline inference. Thus, the model effectiveness was measured on the server, and the efficiency (inference speed) was tested on an edge device.

2) Train and Test Split: To ensure the robustness of our models and their ability to generalize to new subjects, we employed a subjectwise split for training and testing. Our training set consisted of 50 subjects (70%) and our testing set consisted of 22 subjects (30%). To further evaluate the models, we divided the training set into five subjectwise folds and used onefold as a validation set for each of five runs. To account for any variability due to randomness, we repeated this process with five different seeds, resulting in a total of 25 runs using the same test set. The results of these runs were then averaged and the standard deviation was reported.

3) Hyperparameters Used for Model Training: Here, we elaborate on the hyperparameters used for our model training and constructing the RP. We employed a batch size of 512 and the AdamW optimizer [54] with a learning rate of 0.002 for training our model. We chose AdamW as it demonstrated faster convergence and superior performance compared to other optimizer such as Adam. The learning rate was selected through a grid search of values between {0.02, 0.002, 0.0002}. The training process was stopped when the validation loss did not decrease for more than 24 consecutive epochs. Additionally, we set the embedding dimension of the RP to 1, resulting in an image size of 160×160. The Euclidean distance was used as the norm $||\cdot||$ and we set threshold distance ε to be 10% of the maximum distance value.

4) Evaluation Metrics: Three tasks must be evaluated: 1) AC; 2) SE; and 3) BWE. For AC, we used accuracy as our metric. For regression tasks, we report the mean absolute error (MAE), RMSE, and mean absolute percentage error (MAPE). The exact equations for each metrics are listed below

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{7}$$

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}\left(y_i - \hat{y}_i\right)^2}{N}} \tag{8}$$

$$\text{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right|. \tag{9}$$

VI. Results

In this section, we present the experimental results as described in Section V. First, we compare the single-task models and our MTL models in terms of both effectiveness and efficiency. Then, we compare the results of our MobileNetV2 against LSTM and GRU. Additionally, we verify that negative transfer between tasks can occur without careful selection of the MTL loss function. We examine the performance of different MTL loss functions in our experiment. Finally, we visualize the embedding space of our proposed MTL to demonstrate that our model successfully distinguishes the underlying patterns for the three different tasks: 1) AC; 2) SE; and 3) BWE.

A. MobileNetV2-Based MTL Model Versus MobileNetV2-Based Single-Task Models

1) Effectiveness: Table III compares the MobileNetV2-based MTL model against MobileNetV2-based single-task models. Our results show that the MTL model performed equally well or better than single-task models in all AC, SE, and BWE tasks. In the AC task, the MTL model achieved an accuracy of 97.2%, similar to the single-task model's accuracy of 97.1%. In the SE task, the single-task models show better performance compared to the MTL model in all three metrics, MAE, RMSE, and MAPE. For example, the single-task model had a MAE of 0.258 compared to 0.268 in the MTL model. However, the MTL model showed stronger performance in the BWE task in all three metrics. The MTL model had a 6.85 MAE, 8.36 RMSE, and 0.107 MAPE, surpassing the single-task models' 7.27 MAE, 9.04 RMSE, and 0.112 MAPE. In conclusion,

TABLE III
PERFORMANCE COMPARISON BETWEEN SINGLE-TASK AND MULTITASK MODELS

Task	Base Model	Activity	Speed *(km/hr)*			Body Weight *(kg)*		
		Accuracy	MAE	RMSE	MAPE[a]	MAE	RMSE	MAPE
Activity (single-task)	LSTM	94.6 (0.010)	-	-	-	-	-	-
	GRU	93.0 (0.015)	-	-	-	-	-	-
	MobileNetV2	**97.1 (0.003)**	-	-	-	-	-	-
Speed (single-task)	LSTM	-	0.306 (0.060)	0.381 (0.059)	0.057 (0.010)	-	-	-
	GRU	-	0.424 (0.066)	0.491 (0.070)	0.069 (0.010)	-	-	-
	MobileNetV2	-	**0.258 (0.030)**	**0.334 (0.034)**	**0.049 (0.005)**	-	-	-
Body Weight (single-task)	LSTM	-	-	-	-	9.05 (0.769)	11.5 (0.899)	0.137 (0.012)
	GRU	-	-	-	-	8.77 (0.527)	11.0 (0.500)	0.135 (0.010)
	MobileNetV2	-	-	-	-	**7.27 (0.873)**	**9.04 (1.130)**	**0.112 (0.013)**
Multi-task	LSTM	95.4 (0.009)	**0.262 (0.015)**	0.358 (0.022)	0.055 (0.004)	8.48 (1.02)	11.0 (1.34)	0.135 (0.018)
	GRU	95.3 (0.010)	0.263 (0.019)	**0.331 (0.023)**	0.055 (0.005)	8.92 (1.22)	11.4 (1.50)	0.142 (0.022)
	MobileNetV2	**97.2 (0.005)**	0.268 (0.036)	0.349 (0.046)	**0.052 (0.007)**	**6.85 (0.861)**	**8.36 (0.902)**	**0.107 (0.017)**

[a]For MAPE calculation, we excluded experiment subsets that have the speed label as zero due to the outputs becoming arbitrarily high [55]
[b]We utilized the uncertainty-weighted losses for the multi-task scenario for LSTM, GRU, and MobileNetV2
[c]The best metric is in **red bold**, where as the second best metric is colored in **blue bold**

TABLE IV
RP-BASED MOBILENETV2 WITH DIFFERENT MULTITASK LOSS FUNCTIONS

MTL Loss	Activity	Speed *(km/hr)*			Body Weight *(kg)*		
	Accuracy	MAE	RMSE	MAPE	MAE	RMSE	MAPE
Uniform	93.1 (0.023)	0.309 (0.038)	0.390 (0.042)	0.061 (0.008)	6.94 (0.804)	8.60 (1.15)	0.108 (0.014)
RLW[a] (Normal)	92.9 (0.028)	0.316 (0.005)	0.391 (0.051)	0.062 (0.010)	6.93 (1.15)	8.45 (1.36)	0.110 (0.021)
Uncertainty (Ours)	97.2 (0.005)	0.268 (0.036)	0.349 (0.046)	0.052 (0.007)	6.85 (0.861)	8.36 (0.902)	0.107 (0.017)

[a]Random loss weighting

the MobileNetV2-based MTL model demonstrated similar or improved performance compared to the single-task models, with particularly higher performance gains in the BWE task.

2) Efficiency: In Table V, we show the computational cost of our RP-based MobileNetV2 in single and multitask scenarios. Although our MTL model performs three different tasks, the operation only adds up a minor increase in the number of parameters and multiply-accumulate (MAC) operations compared to the most efficient single-task model. The total parameters decreased from a total of 290.1K for all single-task models (AC: 97.3K, SE: 96.4K, BWE: 96.4K) to 97.3K in the MTL model. The model size remains 0.39 mb for all single and multitask models.

Our evaluation of the inference speed on the Jetson Nano showed improved performance for our MTL model compared to the single-task models. The inference speed of a 4 s windowed sample took 112.5 ms (AC: 37.4, SE: 37.5, BWE: 37.6) for single task models on the GPU and 6.39 s (AC: 2.14, SE: 2.12, BWE: 2.13) on the CPU. However, our MTL model takes only 38.0 ms and 2.08 s on both GPU and CPU, leading to a threefold speedup in inference.

Likewise, our MobileNetV2-based MTL model has been found to be both effective and efficient. Its performance is comparable to using multiple single-task models, making it a practical option for real-world applications.

TABLE V
EFFICIENCY COMPARISON BETWEEN SINGLE AND MULTITASK MODELS ON JETSON NANO

# of Task	Task	# of params	MACs *(G)*	Model size *(mb)*
Single-task[a]	Activity	96.7K	0.338	0.39
	Speed	96.4K	0.338	0.39
	BodyWeight	96.4K	0.338	0.39
Multi-task	-	97.3K	0.340	0.39

# of Task	Task	Time to process single sample	
		(GPU/*ms*)	(CPU/*s*)
Single-task	Activity	37.4 (1.09)	2.14 (0.06)
	Speed	37.5 (1.95)	2.12 (0.06)
	BodyWeight	37.6 (1.90)	2.13 (0.05)
Multi-task	-	38.0 (6.85)	2.08 (0.09)

[a]Note that regression tasks (SE & BWE) have an identical model structure
[b]The mean and standard deviation (in brackets) is obtained from five runs

B. RP-Based MobileNetV2 Versus LSTM/GRU

In this section, we compared the RP-based MobileNetV2 with LSTM and GRU in both single-task and multitask settings. We first show that the RP-based MobileNetV2 outperforms in all single-tasks. As shown in Table III, a significant performance gain occurs when the raw input signals are converted to an RP for single-task models of AC, SE, and BWE. For AC, the accuracy is 97.1% for MobileNetV2, whereas the next best performing model is LSTM which has an accuracy of 94.6%. In both SE and BWE, we observe that MobileNetV2 is once again the best performing model compared to LSTM and GRU. In all metrics, MAE, RMSE, and MAPE, MobileNetV2 outperforms GRU and LSTM by quite a margin. Likewise, it is apparent that converting a raw input signal to an RP and processing with the MoibleNetV2 shows enhanced performances for all single-task settings.

In the multitask setting, the MobileNetV2 outperforms LSTM and GRU in both AC and BWE. For AC, we see that the MobileNetV2 has an accuracy of 97.2% while the next best

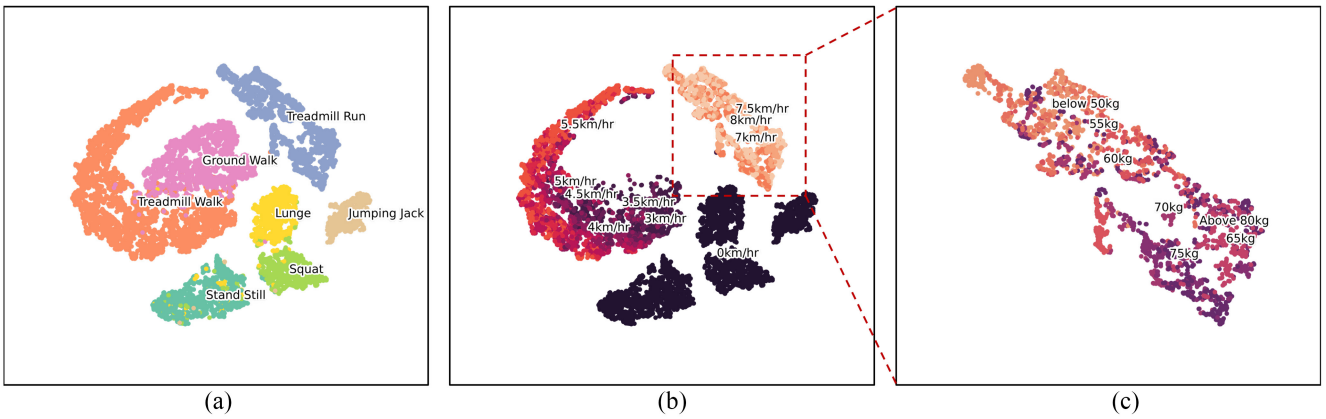

Fig. 4. t-SNE representation of the learned representation of our MTL model is shown in (a)–(c). (a) Embedding space of the AC is displayed and each color represents different types of activities. (b) Embedding space of SE is shown and the predicted speed range is colored in a sequential color range. (c) Embedding space of BWE is displayed, with the note that the body weight is estimated when the user is running on the treadmill.

model is LSTM with 95.4%. For BWE, the MobileNetV2 has an MAE of 6.85 MAE, while LSTM and GRU have scores of 8.48 and 8.92. For SE, we see that all three models have similar MAE, RMSE, and MAPE scores. However, the LSTM and GRU was the best performing model in terms of MAE and RMSE, respectively. The MobileNetV2 had the best MAPE score of 0.052 compared to 0.055 of both LSTM and GRU.

C. MTL Loss Functions

The MobileNetV2-based MTL model was tested with three different MTL loss functions: uniform, RLW, and uncertainty weighted loss. The results, as shown in Table IV, showed that the uncertainty weighted loss function was the most effective, achieving the highest performance in all AC, SE, and BWE metrics. In the AC, the uncertainty weighted loss achieves 97.2% accuracy, while the uniform loss and RLW had a lower accuracy of 93.1% and 92.9%. The uncertainty weighted loss also provided the best results in SE, with an MAE of 0.268, RMSE of 0.349, and MAPE of 0.052. The second best performing loss was the uniform loss which obtained an MAE of 0.309, RMSE of 0.390, and MAPE of 0.061. In BWE, the uncertainty-weighted loss was slightly better than the other two loss functions, with a score of 6.85, 8.36, and 0.107 for MAE, RMSE, and MAPE, respectively.

D. Embedding Space

We show how the embedding space of our MTL result looks like by visualizing the extracted features from the test samples. The features were taken from the last output layer of the shared backbone model. Here, Fig. 4 is the t-SNE representation of the AC (a), the SE (b), and the BWE (c). The three images are identical in shape, and only the relevant subset of experiments for each task was visualized. Note that Fig. 4(c) is a magnified version of the 'treadmill run' class in AC. The labels for each specific AC, SE, and BWE are placed at the median sample of each class. We see that in Fig. 4(a) the embeddings form a unique cluster for each type of activity, with similar activities, such as "Ground Walk" and "Treadmill Walk," "Lunge," and "Squat" set side by side. In Fig. 4(b), we observe that the speed features are placed in sequential order. For Fig. 4(c), the body weights are placed in descending order except for 65 kg. We conjecture that this mismatch can be fixed with a more significant number of samples.

VII. Discussion

This section will discuss two intriguing findings of our study in BWE. First, the performance of BWE is better when running than standing still. Second, our research contributes to commercializing an AI-based BWE model by outperforming the prior BWE research based on a smart insole without utilizing user-related information.

To begin with, one of the interesting aspects of our work is that experiments 9, 10, and 11, the running on the treadmill tasks, worked best for weight estimation compared to other experiments. Generally, a stationary position is thought to be more favorable for weight estimation, as we measure body weight on a scale while standing still. However, a point worth noting is that the FSR sensor measures the relative pressure difference, which is different from how a mechanical scale works. This disparity can be attributed to explaining why estimating the body weight in stationary conditions was not easy, which goes against our intuition. Furthermore, although a user might be standing still, the center of mass in the smart insole keeps changing as it is nearly impossible to stay perfectly stationary [56]. On the contrary, this seemingly imperceptible movement in the body is translated to erratic fluctuation in sensor reading, generating a great deal of noise in the time-series data, making BWE difficult. Paradoxically, we hypothesized that regular activities in a controlled environment, such as running on a treadmill, reduce random noise and create a cyclic pattern in sensor readings. Thus, this "stabilized" condition might make deep learning algorithms easily regress the body weight value.

Consequently, we discovered that while running on a treadmill, the FSR3 sensor values had substantially less noise than other sensors, resulting in a more stable reading of the sensor values. To be specific, we first subdivided samples into five categories depending on the body weight (below 50, 50 to 60, 60 to 70, 70 to 80, and above 80) in each experiment. Then, we calculated the mean of each sample $\mathbf{x}$ and visualized the degree of spread of these mean values by the body weight groups in

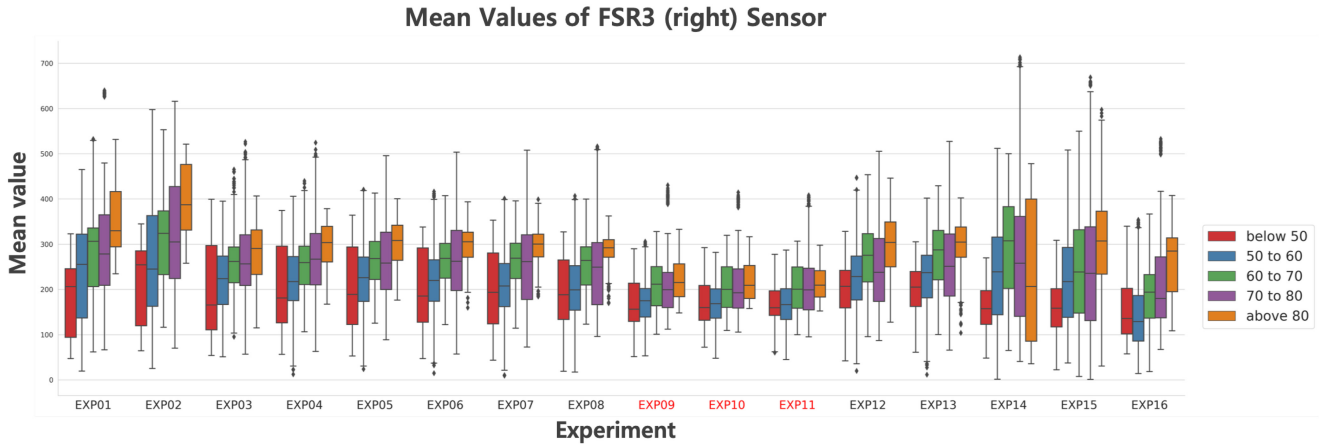

Fig. 5. Distribution of mean values for the FSR3 sensors is depicted in a box plot for each experiment. From the graph, it can be seen that the range of sensor values is relatively low in experiments 9–11, as evidenced by the minimal standard deviation. This indicates that the sensor values in these experiments were more consistent and stable compared to those in other experiments.

every experiment and for each sensor. Intriguingly, we found that in FSR3 sensors, the degree of spread of the mean values per each body weight group was significantly smaller in experiments 9–11 (Fig. 5) compared to the rest, indicating that the noise level of FSR3 sensors drops dramatically when the user is running. In contrast, there was no significant difference in experiments with the rest of the sensors. As to why the FSR3 sensor made a difference, we note that the sensor is embedded at the lateral hill region of both left and right feet, as shown in Fig. 2. Prior studies [57], [58] showed that the plantar pressure is substantially different in the lateral forefoot region per different body weight ranges, implying that the location of the FSR3 sensor is likely to be the critical location in discriminating different body weights.

To further support this assumption, we trained a random forest classifier and a regressor based on several summary statistics (mean, median, min, max, sum, variance, and standard deviation) extracted from the raw sensor data to predict the body weight range of users as classes or as values, respectively. Indeed, the features extracted from the FSR3 sensors are positioned highly (Fig. 6) in both the feature importance of the random forest classifier and regressor. Although this method of showing the importance of features does not exploit the nonlinear interactions between features in deep learning, it seems to be one approach to explain why the "running on the treadmill" experiment works for BWE compared to others.

Finally, we created a commercially feasible BWE model based on a smart insole without compromising user privacy. While several works focus on human AC through wearable sensors, little attention has been given to estimating a user's body weight. Nevertheless, numerous studies demonstrate that monitoring physical activity and body weight changes is vital to maximizing the effect of exercise [59]. Moreover, monitoring body weight is necessary for managing mental and physical health since a sudden change in body weight is a warning sign for multiple disorders, such as depression, obesity, and kidney diseases. Still, keeping track of one's body weight is a proactive action that requires considerable time and effort. Thus, it would be beneficial to automatically estimate

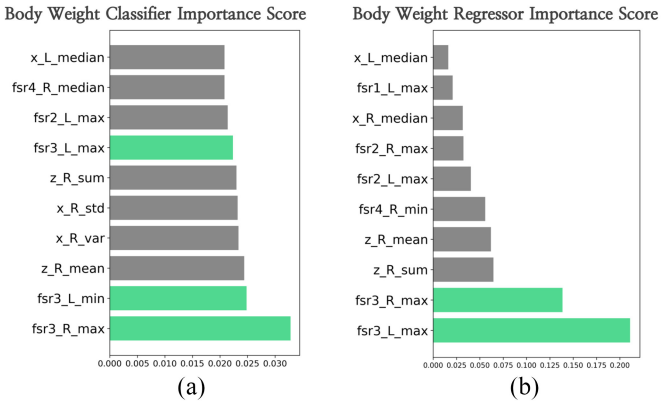

Fig. 6. (a) Feature importance score from the random forest body weight classifier. (b) Feature importance score from the random forest body weight regressor. Both models are trained to predict body weight with several summary statistics extracted from each sensor. Statistics extracted from the FSR3 sensor are colored in green.

body weight with a smart insole that could be worn in our everyday lives. In this work, we are the first to use a deep learning approach to estimate a person's body weight with a commercially viable smart insole. The performance of the proposed BWE model is 8.36 RMSE, which is superior to 10.52 RMSE of [18], which is the only previous study on BWE based on a smart insole. Additionally, whereas most studies on BWE utilized user-related information, such as sex, age, calorie intake, and calorie burn, we demonstrated that it is possible to estimate body weight using only the smart insole's real-time sensor data. Our system is more user-friendly and removes any potential privacy concerns. Most of all, our research opens new possibilities for real-time weight estimation research.

VIII. Conclusion

In this work, we proposed our MTL framework to solve three tasks using a commercial smart insole that addresses wearable-device-related AI healthcare services: AC, SE, and BWE. To build our framework, we created an MTS data set containing a total of 72 different users carrying out 16 different

types of experiments. We showed that our MTL model is lightweight and efficient, with improved or comparable performance compared to the single-task models. Moreover, we show that rather than using the raw sensor signals, converting the data to an image-based RP improved performance in all tasks, especially for BWE. The performance of our MTL framework was verified in various ablation experiments. As for our training strategy, we employed the uncertainty weighing losses to balance the losses incurred in each task. The resulting embedding space of our MTL model makes reasonable clusters within all three tasks.

Finally, we discuss the limitations of our work and the potential future research directions. First, any deep learning model could fail when it faces data distribution that it was not trained on. In our experiments, participants were instructed to walk or run on a treadmill to collect the data for SE and BWE regressor. Unfortunately, the distribution of the data collected under these specific conditions will likely differ from that of data collected in the wild. Second, the BWE works when the user is at a particular running speed range which may limit the applicability. However, our work is the first attempt to develop an MTL model dealing with the three tasks via a commercial smart insole. Thus, future research can focus on collecting the real-world MTL data set containing more than 16 experiments and maximizing the weight estimation performance. Finally, we are confident that our research will bring about new research ideas in the field of HAR and BWE and will serve as helpful for those working with wearable devices.

APPENDIX A
LAYER ABLATION OF MOBILENETV2

Our MobileNetV2 architecture design was experimentally validated through an analysis of the correlation between the number of parameters and the maximum memory allocation for varying numbers of bottleneck layers. As shown in Fig. 7 (Left), we observed that the total number of parameters in our implementation increases exponentially with the number of bottleneck layers. Specifically, it increases from 97.3K for 4 bottlenecks to 350.6K for 5 bottlenecks. On the other hand, Fig. 7 (Right) illustrates that the maximum memory allocated remains relatively consistent between 3 and 4 bottlenecks, but increases from 26K in 4 bottlenecks to 28K in 5 bottlenecks.

We also present the results of our performance evaluation based on different numbers of layers. To simplify the analysis, we use a single metric for each task: AC-Accuracy, SE-RMSE, and BWE-RMSE. Fig. 8 illustrates that for the AC task, the highest accuracy was achieved with four bottlenecks, whereas for both SE and BWE tasks, the best performance was achieved with five bottlenecks, with a marginal difference when using four bottlenecks. These results demonstrate that while there is a slight improvement with the use of additional bottleneck layers, it is not significant enough to justify the exponential increase in parameters and memory allocation, which are crucial considerations for mobile applications. This highlights the reason for our choice to only use the first three and the latest bottlenecks of the original MobileNetV2, as it strikes a balance between performance and resource efficiency.

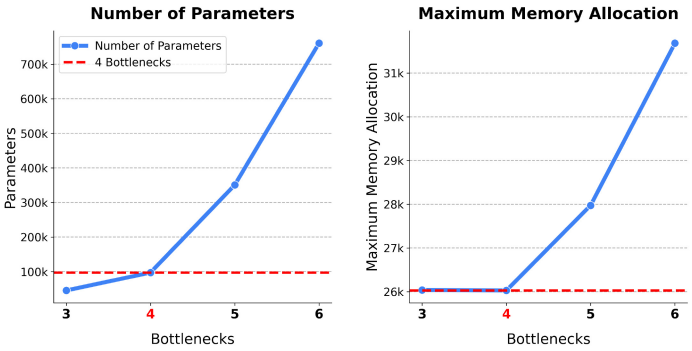

Fig. 7. Figure illustrates the comparison of the number of parameters (Left) and maximum memory allocation (Right) between different numbers of bottlenecks. The left graph shows the exponential increase in the number of parameters as the number of bottlenecks increases, while the right graph illustrates how the maximum memory allocation remains relatively constant between three and four bottlenecks, but increases from 26K in four bottlenecks to 28K in five bottlenecks. This highlights the tradeoff between performance and resource efficiency when using different numbers of bottlenecks.

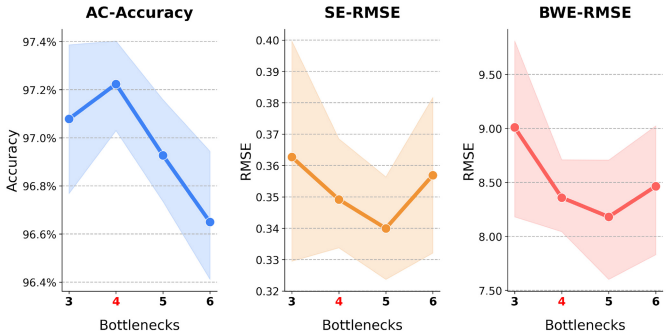

Fig. 8. This figure illustrates the results of our performance evaluation for three different tasks: AC-accuracy (Left), SE-RMSE (Middle), and BWE-RMSE (Right) as a function of the number of bottleneck layers. The left graph shows how the accuracy of the AC task increases with the number of bottlenecks and reaches the highest value with four bottlenecks. The middle and the right graphs, on the other hand, demonstrate that the best performance for both SE and BWE tasks is achieved with five bottlenecks, with a marginal difference when using four bottlenecks. These results show the relationship between the number of bottleneck layers and the performance of the network on different tasks.

TABLE VI
OPTIMIZER COMPARISON RESULTS: ADAMW AND ADAM

Optimizer	Activity	Speed (km/hr)	Body Weight (kg)	Training Epoch
	Accuracy	RMSE	RMSE	
AdamW	**0.972 (0.005)**	0.349 (0.045)	**8.359 (0.901)**	**114.0 (13.34)**
Adam	0.971 (0.006)	**0.342 (0.023)**	8.995 (0.798)	131.6 (34.782)

APPENDIX B
OPTIMIZER COMPARISON RESULTS

Here, we have conducted a comparison study of Adam and AdamW optimization algorithms for our specific MTL task and have found that AdamW results in a smaller number of training epochs while achieving comparable or better MTL results. We kept all other parameters constant and compared Adam (learning rate=0.002) with AdamW (learning rate=0.002) using our MTL model. The results, presented in Table VI, show that AdamW and Adam perform similarly in AC and SE, however, AdamW demonstrated an

improvement in BWE with an RMSE of 8.359(0.901) compared to Adam's 8.995(0.798). Additionally, AdamW had an average training epoch of 114.0(13.34), while Adam took 131.6(34.782) epochs.

ACKNOWLEDGMENT

The authors would like to thank all those who may have contributed to making this work complete. They would also like to thank the participants who participated in our data collection process. A special thanks to the UNIST Data Mining lab members for providing insightful ideas and feedback on this work.

REFERENCES

[1] J. S. Park, J. S. Choi, and D. K. Han, "Platinum nanozyme-hydrogel composite (PtNZHG)-impregnated cascade sensing system for one-step glucose detection in serum, urine, and saliva," *Sens. Actuat. B, Chem.*, vol. 359, May 2022, Art. no. 131585.

[2] M. Huang, I. Gibson, and R. Yang, "Smart chair for monitoring of sitting behavior," in *Proc. Int. Conf. Des. Technol.*, 2017, pp. 274–280.

[3] U. Tholl, K. Forstner, and M. Anlauf, "Measuring blood pressure: Pitfalls and recommendations," *Nephrol. Dialysis Transplantation*, vol. 19, no. 4, pp. 766–770, 2004.

[4] M. E. L. Menshawy, A. Benharref, and M. Serhani, "An automatic mobile-health based approach for EEG epileptic seizures detection," *Expert Syst. Appl.*, vol. 42, no. 20, pp. 7157–7174, 2015.

[5] I. S. Abed, "Lung cancer detection from X-ray images by combined backpropagation neural network and PCA," *Eng. Technol. J.*, vol. 37, no. 5, pp. 166–171, 2019.

[6] G. Cummins, "Smart pills for gastrointestinal diagnostics and therapy," *Adv. Drug Del. Rev.*, vol. 177, Oct. 2021, Art. no. 113931.

[7] J. A. Moral-Munoz, W. Zhang, M. J. Cobo, E. Herrera-Viedma, and D. B. Kaber, "Smartphone-based systems for physical rehabilitation applications: A systematic review," *Assist. Technol.*, vol. 33, no. 4, pp. 223–236, 2021.

[8] N. S. Hoang, Y. Cai, C.-W. Lee, Y. O. Yang, C.-K. Chui, and M. C. H. Chua, "Gait classification for parkinson's disease using stacked 2D and 1D convolutional neural network," in *Proc. Int. Conf. Adv. Technol. Commun. (ATC)*, 2019, pp. 44–49.

[9] M. A. Bencheikh and S. Boukhenous, "A low cost smart insole for diabetic foot prevention," in *Proc. Int. Conf. Appl. Smart Syst. (ICASS)*, 2018, pp. 1–4.

[10] R. Foppen, "Smart insoles: Prevention of falls in older people through instant risk analysis and signalling," M.S. thesis, Dept. Ind. Design Eng., Delft Univ. Technol., Delft, The Netherlands, 2020. [Online]. Available: https://repository.tudelft.nl/islandora/object/uuid:4eccd22d-073f-4411-9ebb-4b8eff50d5a0?collection=education

[11] N. Gkikopoulos, M. Wenger, O. Distler, and M. Becker, "Self-monitoring of the resting heart rate using a fitness tracker smartwatch application leads to an early diagnosis of large vessel vasculitis," *BMJ Case Rep.*, vol. 15, no. 2, 2022, Art. no. e245021.

[12] I. Pernek, K. A. Hummel, and P. Kokol, "Exercise repetition detection for resistance training based on smartphones," *Pers. Ubiquitous Comput.*, vol. 17, no. 4, pp. 771–782, 2013.

[13] F. Gasparetti, L. M. Aiello, and D. Quercia, "Personalized weight loss strategies by mining activity tracker data," *User Model. User Adapt. Interact.*, vol. 30, no. 3, pp. 447–476, 2020.

[14] D. R. Barrow et al., "Exercise prescription for weight management in obese adults at risk for osteoarthritis: Synthesis from a systematic review," *BMC Musculoskeletal Disord.*, vol. 20, no. 1, pp. 1–9, 2019.

[15] I. Machorro-Cano, G. Alor-Hernández, M. A. Paredes-Valverde, U. Ramos-Deonati, J. L. Sánchez-Cervantes, and L. Rodríguez-Mazahua, "PISIoT: A machine learning and IoT-based smart health platform for overweight and obesity control," *Appl. Sci.*, vol. 9, no. 15, p. 3037, 2019.

[16] S. R. Ramamurthy and N. Roy, "Recent trends in machine learning for human activity recognition—A survey," *Wiley Interdiscipl. Rev. Data Min. Knowl. Discov.*, vol. 8, no. 4, p. e1254, 2018.

[17] O. Banos, J.-M. Galvez, M. Damas, H. Pomares, and I. Rojas, "Window size impact in human activity recognition," *Sensors*, vol. 14, no. 4, pp. 6474–6499, 2014.

[18] N. A. Sazonova, R. Browning, and E. S. Sazonov, "Prediction of body-weight and energy expenditure using point pressure and foot acceleration measurements," *Open Biomed. Eng. J.*, vol. 5, pp. 110–115, Dec. 2011.

[19] G. Chakraborty, T. Dendou, D. Kikuchi, and K. Chiba, "How much information could be revealed by analyzing data from pressure sensors attached to shoe insole?" in *Proc. IEEE Int. Instrum. Meas. Technol. Conf.*, 2012, pp. 1963–1967.

[20] T.-H. Kim and Y.-S. Hong, "Prediction of body weight of a person lying on a smart mat in nonrestraint and unconsciousness conditions," *Sensors*, vol. 20, no. 12, p. 3485, 2020.

[21] A. Dobrescu, M. V. Giuffrida, and S. A. Tsaftaris, "Doing more with less: A multitask deep learning approach in plant phenotyping," *Front. Plant Sci.*, vol. 11, p. 141, Feb. 2020.

[22] A. Rago, G. Piro, G. Boggia, and P. Dini, "Multi-task learning at the mobile edge: An effective way to combine traffic classification and prediction," *IEEE Trans. Veh. Technol.*, vol. 69, no. 9, pp. 10362–10374, Sep. 2020.

[23] Y. Charlon, E. Campo, and D. Brulin, "Design and evaluation of a smart insole: Application for continuous monitoring of frail people at home," *Expert Syst. Appl.*, vol. 95, pp. 57–71, Apr. 2018.

[24] D.-Y. Kim, S.-H. Lee, and G.-M. Jeong, "Stack LSTM-based user identification using smart shoes with accelerometer data," *Sensors*, vol. 21, no. 23, p. 8129, 2021.

[25] C. F. Martindale, V. Christlein, P. Klumpp, and B. M. Eskofier, "Wearables-based multi-task gait and activity segmentation using recurrent neural networks," *Neurocomputing*, vol. 432, pp. 250–261, Apr. 2021.

[26] L. Atallah, B. Lo, R. King, and G.-Z. Yang, "Sensor positioning for activity recognition using wearable accelerometers," *IEEE Trans. Biomed. Circuits Syst.*, vol. 5, no. 4, pp. 320–329, Aug. 2011.

[27] K. J. Merry et al., "Classifying sitting, standing, and walking using plantar force data," *Med. Biol. Eng. Comput.*, vol. 59, no. 1, pp. 257–270, 2021.

[28] J. Yang et al., "Smart wearable monitoring system based on multi-type sensors for motion recognition," *Smart Mater. Struct.*, vol. 30, no. 3, 2021, Art. no. 35017.

[29] A. Mannini and A. M. Sabatini, "On-line classification of human activity and estimation of walk-run speed from acceleration data using support vector machines," in *Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc.*, 2011, pp. 3302–3305.

[30] W. Wei, Y. Kaiming, Z. Yu, Q. Yuyang, W. Chenhui, and L. Min, "Walking speed estimation from a wearable insole pressure system embedded with an accelerometer using Bayesian neural network," *J. Eng. Sci. Med. Diagnos. Ther.*, vol. 4, no. 2, 2021, Art. no. 21003.

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

[32] C. A. Ronao and S.-B. Cho, "Human activity recognition with smartphone sensors using deep learning neural networks," *Expert Syst. Appl.*, vol. 59, pp. 235–244, Oct. 2016.

[33] J. Lu and K.-Y. Tong, "Robust single accelerometer-based activity recognition using modified recurrence plot," *IEEE Sensors J.*, vol. 19, no. 15, pp. 6317–6324, Aug. 2019.

[34] W. S. Low et al., "Lower extremity kinematics walking speed classification using long short-term memory neural frameworks," *Multimedia Tools Appl.*, vol. 82, pp. 9745–9760, Mar. 2022.

[35] V. D. R. Seethi and P. Bharti, "CNN-based speed detection algorithm for walking and running using wrist-worn wearable sensors," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, 2020, pp. 278–283.

[36] O. Barut, L. Zhou, and Y. Luo, "Multitask LSTM model for human activity recognition and intensity estimation using wearable sensor data," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8760–8768, Sep. 2020.

[37] C.-W. Lin, T.-C. Wen, and F. Setiawan, "Evaluation of vertical ground reaction forces pattern visualization in neurodegenerative diseases identification using deep learning and recurrence plot image feature extraction," *Sensors*, vol. 20, no. 14, p. 3857, 2020.

[38] J.-P. Eckmann, S. O. Kamphorst, and D. Ruelle, "Recurrence plots of dynamical systems," *Europhys. Lett.*, vol. 4, no. 9, p. 973, Nov. 1987. [Online]. Available: https://iopscience.iop.org/article/10.1209/0295-5075/4/9/004

[39] N. Hatami, Y. Gavet, and J. Debayle, "Classification of time-series images using deep convolutional neural networks," in *Proc. 10th Int. Conf. Mach. Vis. (ICMV)*, 2018, pp. 242–249.

[40] B. Bertalanič, M. Meža, and C. Fortuna, "Resource-aware time series imaging classification for wireless link layer anomalies," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Feb. 17, 2022, doi: 10.1109/TNNLS.2022.3149091.

[41] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4510–4520.
[42] K. Kehelella et al., "Vision transformer with convolutional encoder–decoder for hand gesture recognition using 24-GHz Doppler radar," *IEEE Sens. Lett.*, vol. 6, no. 10, pp. 1–4, Oct. 2022.
[43] Y. Tang, L. Zhang, H. Wu, J. He, and A. Song, "Dual-branch interactive networks on multichannel time series for human activity recognition," *IEEE J. Biomed. Health Inform.*, vol. 26, no. 10, pp. 5223–5234, Oct. 2022.
[44] Z. Lu, W. Lv, Y. Cao, Z. Xie, H. Peng, and B. Du, "LSTM variants meet graph neural networks for road speed prediction," *Neurocomputing*, vol. 400, pp. 34–45, Aug. 2020.
[45] L. Romo, J. Zhang, K. Eastin, and C. Xue, "Short-term traffic speed prediction via machine learning," in *Proc. 15th Int. Conf. Green Pervasive Cloud Comput. (GPC)*, Xi'an, China, Nov. 2020, pp. 31–42.
[46] M. Crawshaw, "Multi-task learning with deep neural networks: A survey," 2020, *arXiv:2009.09796*.
[47] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7482–7491.
[48] A. Paszke et al., "Automatic differentiation in PyTorch," in *Proc. 31st Conf. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 1–4.
[49] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[50] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014, *arXiv:1412.3555*.
[51] S. Wang, J. Cao, and P. S. Yu, "Deep learning for spatio-temporal data mining: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 8, pp. 3681–3700, Aug. 2022.
[52] C. Pelletier, G. I. Webb, and F. Petitjean, "Temporal convolutional neural network for the classification of satellite image time series," *Remote Sens.*, vol. 11, no. 5, p. 523, 2019.
[53] Y. Liu et al., "Ensemble spatiotemporal forecasting of solar irradiation using variational Bayesian convolutional gate recurrent unit network," *Appl. Energy*, vol. 253, Nov. 2019, Art. no. 113596.
[54] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2017, *arXiv:1711.05101*.
[55] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Nov. 2011.
[56] N. Yamada, "Chaotic swaying of the upright posture," *Human Movement Sci.*, vol. 14, no. 6, pp. 711–726, 1995.
[57] A. M. Dowling, J. R. Steele, and L. A. Baur, "What are the effects of obesity in children on plantar pressure distributions?" *Int. J. Obesity*, vol. 28, no. 11, pp. 1514–1519, 2004.
[58] M. Birtane and H. Tuna, "The evaluation of plantar pressure distribution in obese and non-obese adults," *Clin. Biomech.*, vol. 19, no. 10, pp. 1055–1059, 2004.
[59] D. E. Alley, L. Ferrucci, M. Barbagallo, S. A. Studenski, and T. B. Harris, "A research agenda: The changing relationship between body weight and health in aging," *J.. Gerontol. Ser. A, Biol. Sci. Med. Sci.*, vol. 63, no. 11, pp. 1257–1259, 2008.

Jaeho Kim received the B.S. degree in management engineering and computer science from Ulsan National Institute of Science and Technology, Ulsan, Republic of Korea, in 2020, where he is currently pursuing the Ph.D. degree in artificial intelligence with the Graduate School of Artificial Intelligence.

His research interests include deep learning with application and time-series problems.

Hyewon Kang received the B.S. degree in international office administration & psychology from Ewha Womans University, Seoul, Republic of Korea, in 2016. She is currently pursuing the M.S. degree in industrial engineering with the Department of Industrial Engineering, Ulsan National Institute of Science and Technology, Ulsan, Republic of Korea.

Her research interest is graph-based time-series data analysis.

Jaewan Yang received the M.S. degree in electrical and electronic computer engineering from Ulsan University, Ulsan, Republic of Korea, in 2019.

He is currently a Researcher with Gilon Inc., Seoul, Republic of Korea. His research interests are in the areas of data science, artificial intelligence, and signal processing.

Haneul Jung received the M.S. degree in biomedical engineering from Yonsei University, Seoul, Republic of Korea, in 2020.

He is currently a Researcher with Gilon Inc., Seoul. His research interests are in the areas of rehabilitation-engineering, motion analysis, and artificial intelligence.

Seulki Lee (Member, IEEE) received the B.E. degree in electrical and computer engineering from the University of Seoul, Seoul, South Korea, in 2009, and the M.S. and Ph.D. degrees in computer science from the University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, in 2018 and 2021, respectively.

He is an Assistant Professor with the Department of Computer Science and Engineering and Artificial Intelligence Graduate School, Ulsan National Institute of Science and Technology, Ulsan, Republic of Korea, where he leads the Embedded Artificial Intelligence Lab.

Junghye Lee received the B.S. and Ph.D. degrees from the Department of Industrial and Management Engineering, POSTECH, Pohang, Republic of Korea, in 2012 and 2017, respectively.

She is an Assistant Professor with the Technology Management, Economics and Policy Program, Seoul National University (SNU), Seoul, South Korea. Before joining SNU, she was working as an Associate Professor with Ulsan National Institute of Science and Technology, Ulsan, Republic of Korea. She worked as a Postdoctoral Researcher with the Biomedical Informatics Department, University of California at San Diego, La Jolla, CA, USA. Her main interests include machine learning and deep learning with applications, especially deep representation-learning-based predictive modeling, and privacy-preserving federated learning.