ORIGINAL RESEARCH
published: 14 July 2022
doi: 10.3389/fcomp.2022.925108

Few-Shot and Weakly Supervised Repetition Counting With Body-Worn Accelerometers

Yuuki Nishino, Takuya Maekawa and Takahiro Hara*

Department of Multimedia Engineering, Graduate School of Information Science and Technology, Osaka University, Suita, Japan

This study investigates few-shot weakly supervised repetition counting of a human action such as workout using a wearable inertial sensor. We present WeakCounterF that leverages few weakly labeled segments containing occurrences of a target action from a target user to achieve precise repetition counting. Here, a weak label is defined to specify only the number of repetitions of an action included in an input data segment in this study, facilitating preparation of datasets for repetition counting. First, WeakCounterF leverages data augmentation and label diversification techniques to generate augmented diverse training data from weakly labeled data from users other than a target user, i.e., source users. Then, WeakCounterF generates diverse weakly labeled training data from few weakly labeled training data from the target user. Finally, WeakCounterF trains its repetition counting model composed of an attention mechanism on the augmented diversified data from the source users, and then fine-tunes the model on the diversified data from the target user.

Keywords: wearable sensor, repetition counting, few-shot learning, weakly supervised learning, human action

Edited by:
Katia Vega,
University of California, Davis,
United States

Reviewed by:
Shadi Abudalfa,
University College of Applied Sciences, Palestine
Xiaopeng Niu,
Peking University, China

****Correspondence:***
Takuya Maekawa
maekawa@ist.osaka-u.ac.jp

Specialty section:
This article was submitted to Mobile and Ubiquitous Computing, a section of the journal Frontiers in Computer Science

Received: *21 April 2022*
Accepted: *21 June 2022*
Published: *14 July 2022*

Citation:
Nishino Y, Maekawa T and Hara T (2022) Few-Shot and Weakly Supervised Repetition Counting With Body-Worn Accelerometers. Front. Comput. Sci. 4:925108. doi: 10.3389/fcomp.2022.925108

1. INTRODUCTION

Recent proliferation of wearable sensor devices, such as smart watches, paved the way for individuals to track their health and work efficiently. Specifically, in the wearable computing research community, real-world applications based on body-worn inertial sensors in sports and industrial environments have been actively studied (Aehnelt et al., 2014; Guo et al., 2017).

In this study, we focus on repetition counting methods for an action using body-worn inertial sensors because they are effective in sports and industrial environments that need to record repetition data of workouts for exercise management and factory worker logs for performance verification of predefined tasks, respectively (Lukowicz et al., 2004; Maekawa et al., 2016; Xia et al., 2019, 2020; Morales et al., 2022). Prior studies on repetition counting rely on supervised learning such as neural networks. However, these methods require costly data annotation of target actions as depicted in **Figure 1**, where a label specifying the start time of each occurrence of the target action and another specifying a short segment containing repetitions of the target action are required.

To address these issues, we have proposed the so-called WeakCounter (Nishino et al., 2021), which is a weakly supervised method developed for repetition counting. WeakCounter assumes that an input, namely, a three-axis acceleration segment from a smartwatch, and a weak label, which is defined to specify *only* the number of repetitions of the target action included in the input segment, are given (**Figure 1**). Unlike prior studies, WeakCounter does not require the two costly labels. An attention-based neural network is used in WeakCounter, which is equipped with a

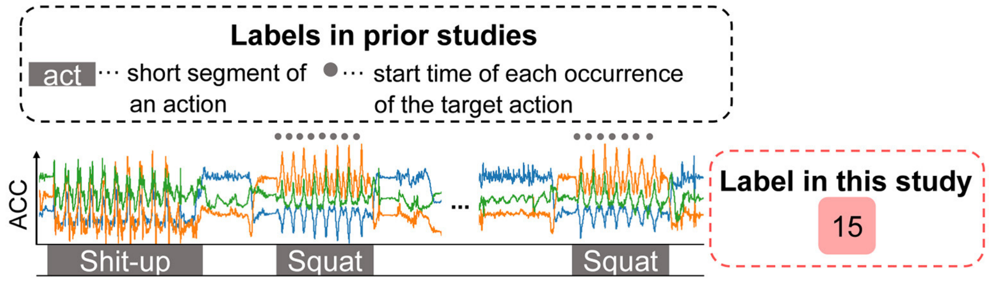
Labels in prior studies
act ··· short segment of an action
● ··· start time of each occurrence of the target action
ACC
Shit-up
Squat
Squat
Label in this study
15

FIGURE 1 | Example labels for supervised methods used in prior studies and our study. This example assumes to count the number of repetitions of the squat action. Our study requires only a numerical value for the input segment, i.e., the number of repetitions of the squat action.

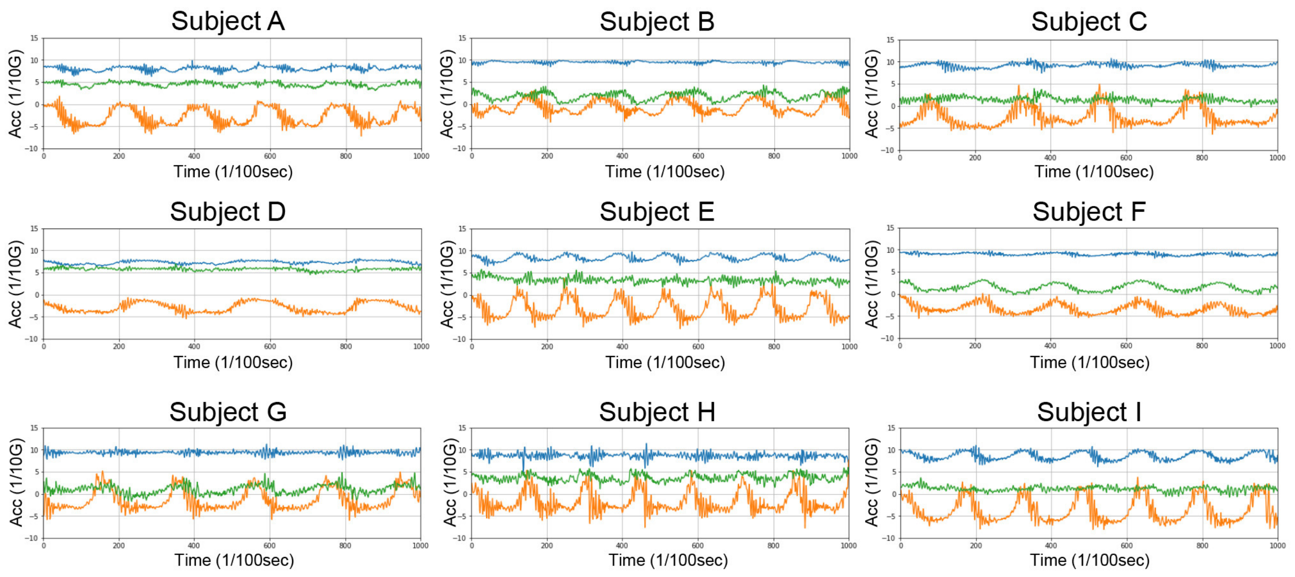

FIGURE 2 | Smartwatch acceleration sensor data regarding push-up actions from nine participants, indicating the individual differences in sensor data. Blue, orange, and green lines indicate x-, y-, and z-axis sensor data, respectively.

detection block that detects the occurrences of the target action *via* an attention mechanism and a counting block that counts the number of detected occurrences.

WeakCounter was evaluated through the leave-one-user-out cross validation framework where labeled sensor data from a test user is unavailable. Therefore, the different physical traits of participants, which causes a difference in acceleration waveforms of workouts as depicted in **Figure 2**, can deteriorate the performance of the weakly supervised method. To address this issue, this study proposes a few-shot weakly supervised method for repetition counting, which is called WeakCounterF. Therefore, we train a neural network for repetition counting on few test user training data with weak labels. This study assumes that a training system asks a test user (target user) to perform a target action repeatedly. For example, after starting acceleration data collection, the user performs three push-ups, then, the data collection stops. Note that the number of repetitions, namely, three, is requested by the system before the collection starts. Because the user does not need to label the acceleration data, that is, locating each occurrence of the actions in the data, this approach significantly reduces the cost related to data collection and annotation. This study leverages few weakly labeled sensor data segments for each target action to achieve few-shot repetition counting.

Few-shot weakly supervised learning of repetition counting is challenging because (i) waveforms of a target action can vary even when the same test user performs it, (ii) a test sensor data segment measured from the test user actions can contain waveforms unrelated to a target action, for example, other actions such as resting/drinking water activities and other workouts, making it difficult to construct a robust counting model from few weakly labeled data, (iii) training a counting model on few weakly labeled segments with limited variety from the test user is difficult and deteriorates the robustness of the model.

To address these issues, we first train a counting model on weakly labeled sensor data obtained from multiple users other than the test user (source users), which enables constructing a robust counting model. Here, we use an attention-based neural network to specifically focus on a short sensor data segment corresponding to the target action. Then, we leverage few weakly labeled segments obtained from the test user to adapt the trained counting model to the test user actions by fine-tuning the model. To improve the fine-tuned model robustness, we extract short segments each corresponding to individual occurrence of the target action from the weakly labeled segment from the test user by leveraging the correlation among the occurrences. Then, we generate diverse training segments with different labels, i.e., the number of repetitions, by concatenating the extracted short segments for fine-tuning using data augmentation, enabling to construct a robust counting model.

The contributions of our research are as follows: (i) To the best of our knowledge, this is the first application of few-shot weakly supervised learning for repetition counting. (ii) The proposed method is designed to construct a robust counting model using few weakly labeled sensor data obtained from a test user. (iii) The effectiveness of our method is investigated using real sensor data collected from nine participants.

In the rest of this paper, we first review studies on repetition counting, and transfer learning and few-shot learning for activity recognition. We then present our proposed method for few-shot repetition counting, which is called WeakCounterF, and evaluate our method using sensor data collected in real-world environments.

2. RELATED WORK

2.1. Repetition Counting Using Acceleration Data

Repetition counting methods using time-series sensor data can be categorized in two, namely, parametric and supervised methods. Parametric methods rely on predefined parameters, for example, the sensor data magnitudes corresponding to a target action execution and time interval between two consecutive occurrences of the target action (Seeger et al., 2011; Skawinski et al., 2014, 2019; Sundholm et al., 2014; Bian et al., 2019). These parameters should be selected a priori for each action type. Recently developed supervised methods rely on neural networks, such as recurrent networks and 1D convolution networks, for time series data (Pernek et al., 2013; Mortazavi et al., 2014; Soro et al., 2019). Even though the supervised methods do not require predefined parameters, the occurrence labels are significantly costly to prepare (**Figure 1**). Contrarily, our method is designed as an end-to-end architecture for weakly supervised learning.

WeakCounter (Nishino et al., 2021), which is a method proposed in our prior study, trains a weakly supervised counting model on labeled data obtained from users different from the test user because obtaining labeled data from the test user is costly. However, this approach suffers from differences in observed sensor signals regarding the same workout type between different users as depicted in **Figure 2**, which results in inaccurate counting performance. In contrast, our current study leverages few-shot weakly supervised learning to alleviate the sensor data differences regarding different users.

2.2. Transfer and Few-Shot Learning for Acceleration-Based Activity Recognition

Since wearable human activity recognition relying on deep learning requires a considerable amount of labeled sensor data, transfer learning techniques for wearable human activity recognition have been actively studied by wearable and ubiquitous computing research communities (van Kasteren et al., 2010; Hu et al., 2011; Wang et al., 2018). Many of these studies trained deep learning-based activity recognition models on labeled sensor data acquired from source users, which were then adapted to a target user by employing limited training data from the target user. For example, Wang et al. (2018) proposed a recognition network trained by minimizing the maximum mean discrepancy (MMD) of data points that belong to the same class between different domains.

Few-shot learning techniques, which transfer knowledge from related tasks, have been studied to efficiently train activity recognition models with limited training data for a target task (Feng and Duarte, 2019; Deng et al., 2020). For example, Feng and Duarte (2019) proposed a few-shot learning-based activity recognition method that determines the initial parameters of the recognition model of the target domain by employing the similarity between the target and source domains. WeakCounterF employs few-shot learning to count the actions of a target user by fine-tuning a pretrained model that has been trained on data from source users. However, unlike the aforementioned models, WeakCounterF relies on limited weakly labeled training data, which deteriorates the performance of the fine-tuning process. To address this issue, we extract short segments, each corresponding to the individual occurrence of a target action, and generate diverse training data.

3. WEAKCOUNTERF: FEW-SHOT WEAKLY SUPERVISED REPETITION COUNTING

3.1. Preliminaries

Given a three-axis acceleration data segment from a wearable sensor such as a smart watch that contains occurrences of a target action such as push-up and actions different from the target action such as other types of workout and actions of drinking water, the goal of this study is to estimate the number of repetitions of the target action included in the segment. To achieve this, we prepare training segments containing these actions. However, because collecting a sufficient amount of training segments from a target user is costly, we leverage training segments collected from other users (source users). In addition, to address the problem of sensor data difference between different users, we ask the test user to perform the target action few times and use the collected data with a weak label y_t, i.e., the number of repetitions, as additional training data. When we have multiple target actions such as push-up, squat, and, sit-up, we prepare a weakly labeled segment for each target action. In such case, we also train a repetition counting model for each target action.

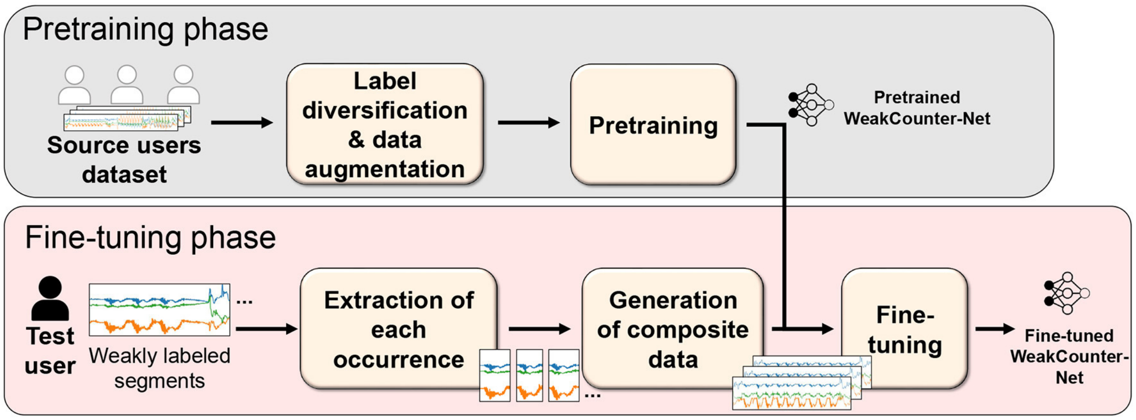
Pretraining phase
Source users dataset
Label diversification & data augmentation
Pretraining
Pretrained WeakCounter-Net
Fine-tuning phase
Test user
Weakly labeled segments
Extraction of each occurrence
Generation of composite data
Fine-tuning
Fine-tuned WeakCounter-Net

FIGURE 3 | Overview of few-shot weakly supervised repetition counting method.

In summary, for each target action, a few weakly labeled acceleration segments from both a test user and the source users are provided as training data. A weakly labeled segment contains occurrences of the target action, and a weak label specifying the number of occurrences of the target action is associated with the segment. Note that a weakly labeled segment collected from the source user can contain actions other than the target action.

3.2. Method Overview

An overview of WeakCounterF is shown in **Figure 3**. The proposed method comprises two main phases: pretraining and fine-tuning. In the pretraining phase, we pretrain an attention-based neural network for repetition counting of a target action, WeakCounter-Net, on source users' data in advance (Nishino et al., 2021). When we have multiple target actions, we prepare a network for each target action. To train a robust pretrained network, we generate diverse training data from the source users' sensor data by employing data augmentation and label diversification techniques. The counting network called WeakCounter-Net is designed based on a counting method used by humans, to achieve robustness.

The fine-tuning phase is composed of *Generation of Fine-tuning Dataset* and *Fine-tuning WeakCounter-Net*. In *Generation of Fine-tuning Dataset*, we construct a dataset that is used to adapt the pretrained network to the target user from weakly labeled data segments collected from the target user. We first extract a short segment corresponding to each occurrence of a target action from the weakly labeled segment. By replicating and concatenating the extracted short segments of the occurrences, we create a variety of composite segments, with each of them composed of multiple occurrences of the target action. For *Fine-tuning WeakCounter-Net*, the pretrained model was fine-tuned on the generated dataset.

3.3. Pretraining Phase

3.3.1. Label Diversification and Data Augmentation

We first perform label diversification that increases the variation of training labels. The basic idea of this method is simple. When we concatenate several data segments, the ground truth label (i.e., no. of repetitions) of the concatenated segment is given by the sum of the ground truth labels of the original segments. Based on this idea, we can easily generate diverse weakly labeled data.

We perform label diversification for each pair of segments from a pool of source users' weakly labeled training segments. Assuming that a pair consisting of segment s_1 with label y_1 and segment s_2 with label y_2 is given. We concatenate s_1 with s_2 and attach a label $y_1 + y_2$ to the concatenated segment. (s_1 and s_2 can also be the same segment). This approach enables the generation of ${}_{N_t}C_2 + N_t$ new labeled segments from the original N_t training segments because we generate a new labeled segment from each pair of training segments. Here, ${}_nC_r = \frac{n!}{(n-r)!r!}$.

Subsequently, for each of the labeled segments, we perform data augmentation to generate N_a diverse segments in terms of the order of actions performed, amplitude of the action, speed of the action, and sensor pose, thereby preventing overfitting. **Figure 4** shows an overview of the data augmentation process. We first split the segments into subsegments because we shuffle those subsegments in the following procedure. Here, the target action should not be performed simultaneously at the breakpoint of the split. Therefore, we compute the variance within a sliding time window and select the center of the window with a variance smaller than the threshold as the breakpoint. We then iterate the following procedure N_a times. For each subsegment, we perform the following operations according to (Morris et al., 2017; Yu et al., 2017):

- Scaling: We change the scale of the sensor data magnitude by multiplying a random value sampled from the uniform distribution $U(1 - c_s, 1 + c_s)$ for each subsegment.
- Time warping: We elongate or contract each subsegment by up/down-sampling the subsegment using the linear interpolation such that the length of the converted subsegment becomes $l \cdot U(1 - c_t, 1 + c_t)$, where l is the length of the original subsegment.

We then shuffle and concatenate the subsegments to generate a complete segment. Finally, we randomly shift the acceleration

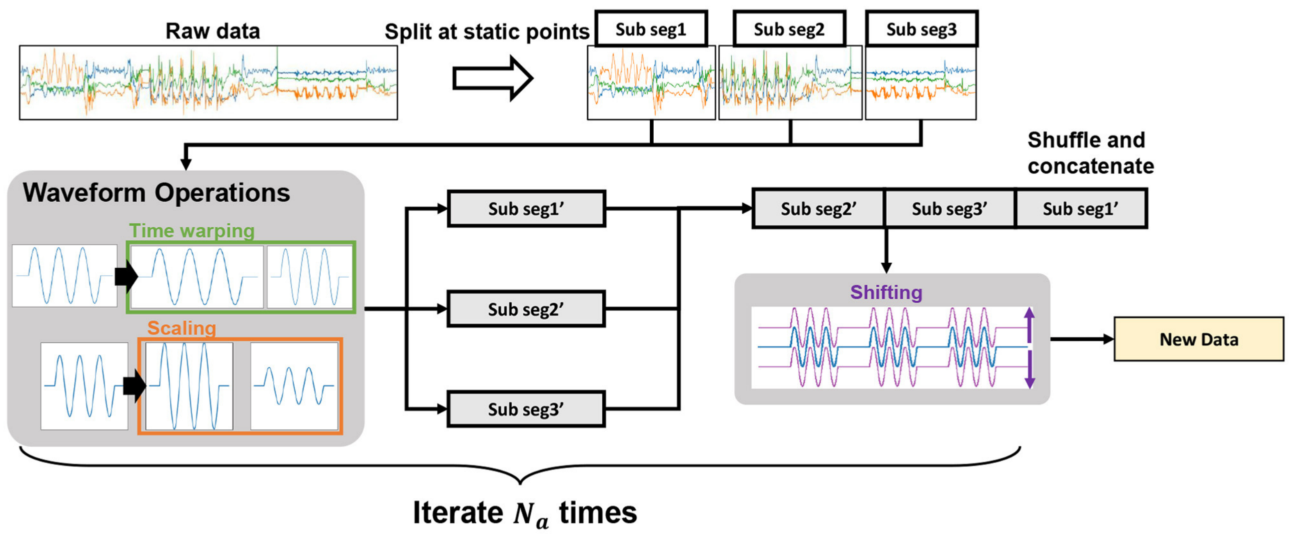

FIGURE 4 | Procedure of data augmentation.

values for each axis to achieve robust counting against the differences in sensor poses by adding a bias value sampled from the uniform distribution $U(-c_b, c_b)$.

3.3.2. WeakCounter-Net

When humans count the number of apples on a table, they first identify each apple and then count the number of identified objects (i.e., apples). Similarly, WeakCounter-Net is designed to be equipped with a detection block that detects the occurrences of the target action using an attention mechanism and a counting block that counts the number of detected occurrences. The detection block outputs a 1D time-series that exhibits a peak at the time of occurrence of the target action, and the output is fed into the counting block. As a result, we can train the counting block to count the number of peaks within the 1D time-series. This architecture is expected to robustly count an arbitrary number of occurrences even if the variation in the training labels is limited.

As shown in **Figure 5**, the model consists of detection and counting blocks. An attention mechanism (Xiao et al., 2015; Zeng et al., 2018; Yu et al., 2019) is introduced into the detection block to focus on the target action because unrelated actions can be included within an input segment. First, we perform feature extraction with 1D convolution layers to output D_f-dimensional time-series $\boldsymbol{F} \in \mathbb{R}^{D_f \times L}$, where L is the data length. Then, we compute attention by $\boldsymbol{a} = \text{sigmoid}[\text{Conv2}(\boldsymbol{F})]$, where $\text{Conv2}(\cdot)$ is two 1D convolution layers with dropout in the attention mechanism, as shown in **Figure 5**.

Here, $\boldsymbol{a} \in \mathbb{R}^{1 \times L}$ indicates the importance (i.e., attention) of each data point in $\boldsymbol{F}$. After performing element-wise multiplication of the attention time series by $\boldsymbol{F}$, by summing the result over the axis of dimension (i.e., D_f), the detection block outputs a 1D time series (i.e., $1 \times L$) that exhibits a peak at the occurrence of the target action (see bottom of **Figure 5**). The counting block processes the 1D time series with 1D convolution, max pooling, and fully connected layers to output an estimate.

We train WeakCounter-Net on weakly labeled data from source users by employing the Adam optimizer (Kingma and Ba, 2014) to minimize the mean squared error between the estimates and the ground truth. When we train WeakCounter-Net for a certain action, such as push-up, we employ segments containing push-up actions with weak labels for the action collected from the source users for the training.

3.4. Fine-Tuning Phase

Here, we fine-tune the pretrained WeakCounter-Net model by employing limited training data from the target user. As mentioned above, a few sensor data segments containing a few repetitions of the target action collected from the test user are provided. With these segments, we generate a diverse dataset and then fine-tune the pretrained model on the dataset.

3.4.1. Generation of Fine-Tuning Dataset

3.4.1.1. Extraction of Each Occurrence of Target Action

In this process, we first extract each occurrence of the target action from the weakly labeled segment. Note that the segment has only a label that specifies the number of repetitions of the target action included in the segment. By employing the weak label, we extract each occurrence of the target action, that is, determine the start time and length of each occurrence. The upper part of **Figure 6** shows an example sensor data segment collected from a test user. As shown in the figure, we can identify three occurrences of the target action "sit-up." However, we can also see waveforms unrelated to the target action at the beginning and end of the segment, which correspond to actions related to the start and end of the data collection. In this method, we first roughly determine the start time and length of each occurrence by employing auto-correlation and then determine the exact start time and length using an optimization technique.

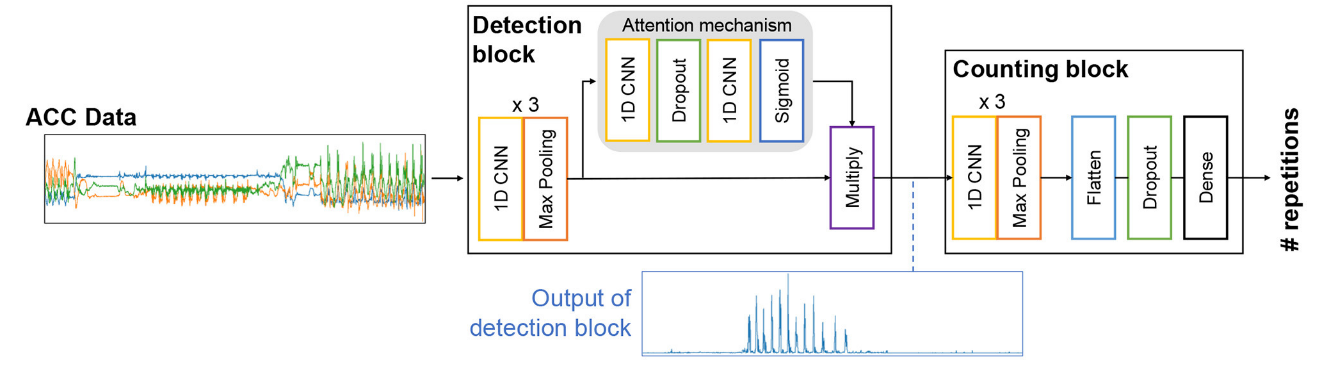

FIGURE 5 | Architecture of WeakCounter-Net.

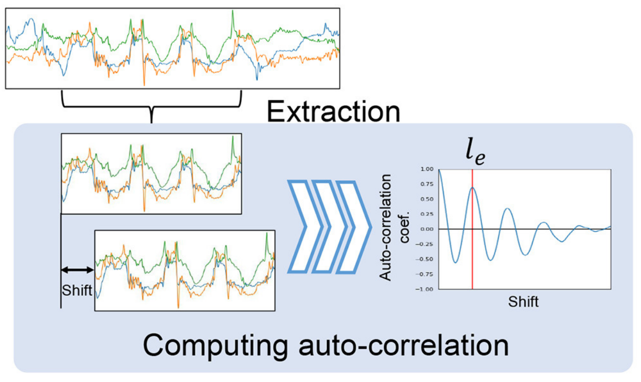

FIGURE 6 | Determining the length of the target action using auto-correlation.

1. *Determining candidates of the length of target action*: Since the segment contains repetitions of the target action, we can estimate the duration (length) of the target action by computing the autocorrelation of the segment. However, as illustrated in **Figure 6**, waveforms unrelated to the target action are included in the segment, making it difficult to estimate the duration by simply using the autocorrelation. Therefore, we extract a subsegment with the length of l/n from the segment using a sliding window, where l is the length of the segment, and then calculate the autocorrelation for each subsegment. Consequently, we can expect that some subsegments that do not contain unrelated waveforms are extracted. **Figure 6** shows an example extracted subsegment and computation of the auto-correlation coefficients by calculating the Pearson correlation between the original subsegment and shifted subsegment while changing the shift value. As shown in **Figure 6**, the shift value of l_e yields the maximum peak of the correlation coefficient, indicating that the duration of the target action is l_e. We calculate the autocorrelation coefficients for the subsegments by changing n and take the shift values yielding the top-N_l peaks of autocorrelation coefficients as candidates for the length of the target action. The detailed procedure is presented in **Algorithm 1**.

Algorithm 1 Determining candidates of the length of target action.

```
Input: $ACC, y_t$ /* input three-axis acceleration data and ground truth label */
Output: $L_{can}$ /* candidates of the length of target action */
    /* pre-processing */
 1: $ACC_s$ ← Scaling each dimension of $ACC$ with IQR
 2: $ACC_p$ ← Smoothing $ACC_s$ with low pass filter and combine three dimensions with RMS
 3: $V_{can} = [], I_{can} = []$
 4: /* extracting sub-segments with length $l/n$ */
 5: for $n = 1 \dots n_s$ do
 6:     $l_s \leftarrow l/n$
 7:     /* sliding sub-segment with stride $w_s$ */
 8:     for $j = 0, w_s, \dots l - l_s$ do
 9:         /* calculate auto-correlation of sub-segment and find max peak */
10:         $s_{sub} \leftarrow ACC_p[j : j + l_s]$
11:         $co_{auto}$ ← Calculate sequence of auto-correlation for $s_{sub}$
12:         $V_{peak}, I_{peak}$ ← Value and index of each detected peak in $co_{auto}$
13:         $v_{max} \leftarrow \max(V_{peak}), i_{max} \leftarrow \mathrm{argmax}(V_{peak})$
14:         $V_{can}$.append($v_{max}$), $I_{can}$.append($i_{max}$)
15:     end for
16: end for
17: $O_{sort}$ = argsort($V_{can}$) /* indices of $V_{can}$ are sorted in descending order of peak values */
18: $L_{can} \leftarrow I_{can}[O_{sort}[: N_l]]$ /* extract top-$N_l$ peaks in $O_{sort}$ */
```

2. *Determining the start time of the first occurrence*: We then roughly estimate the start time of the first occurrence of the target action by employing each candidate of the estimated length l_e in the above procedure. Note that the weak label y_t (i.e., the number of occurrences of the target action) of the segment is given, and the action is assumed to be continually performed. Therefore, when we assume that the start time

of the first occurrence is t_{s1} and the number of occurrences is y_t, the sensor data similarity among $s(t_{s1}, l_e)$, $s(t_{s1} + l_e, l_e)$, $s(t_{s1} + 2l_e, l_e), \ldots, s(t_{s1} + (y_t - 1)l_e, l_e)$ should be high, where $s(t, l)$ shows a short data segment starting at time t with a length of l, as shown in **Figure 7A**. In our method, we scan the segment to find t_{s1} that yields the minimum dynamic time warping (DTW) distance among the short segments, which is calculated as follows:

$$\sum_{i,j \in N_{y_t}} \mathrm{DTW}(s(t_{s1} + i \cdot l_e, l_e), s(t_{s1} + j \cdot l_e, l_e)), \qquad (1)$$

where $\mathrm{DTW}(\cdot,\cdot)$ is the DTW distance between two data segments, and N_{y_t} is a set of integer values larger than or equal to zero and smaller than y_t. **Figure 7A** exhibits example identified occurrences based on estimated start time and duration. The detailed procedure is presented in **Algorithm 2**.

3. *Determining the start time and length of each occurrence*: In the above procedure, we roughly obtained the start time of the first occurrence and the duration. Note that we assumed that the duration of all occurrences are identical. Here, using an optimization technique, we obtain the start time and duration of each occurrence because the duration of each occurrence can be different, and there can be a short blank between two consecutive occurrences. The objective function of the optimization is

$$\sum_{i,j \in N_{y_t}} \mathrm{DTW}(s(t_{s_i}, l_{s_i}), s(t_{s_j}, l_{s_j})), \qquad (2)$$

where t_{s_i} is the start time, and l_{s_i} is the length of the ith occurrence. We find t_{s_i} and l_{s_i} that minimize the objective function by using the L-BFGS-B optimizer (Zhu et al., 1997). During the optimization, the following restrictions are introduced to obtain final results that do not deviate from the initial estimates obtained in the previous procedure. (i) The absolute time difference between the estimated start time of the ith occurrence and the initial start time estimated in the previous procedure should be smaller than l_e. (ii) The difference between the estimated lengths of the ith occurrence should be similar ($<0.2\ l_e$). **Figure 7B** shows example identified occurrences by the optimization.

3.4.1.2. Generation of Composite Data

In the above procedure, we obtained estimates of the start time and length of each occurrence. By replicating and concatenating the segments corresponding to the occurrences, we generate training data to fine-tune the pretrained model. Here, we generate N_f weakly labeled segments for each label value y_i, where $1 \leqq y_i \leqq y_{max}$. We generate a weakly labeled segment with label y_i as follows:

1. We randomly select y_i occurrences (short data segments) from a weakly labeled segment from the target user. Recall that we know the start and end times of each occurrence in the weakly labeled segment from the target user in the above procedure. To improve the robustness of the counting model, we randomly change the start and end times of an occurrence by adding r to the start/end time when we extract the occurrence, where $-0.1l_e < r < 0.1l_e$.
2. We concatenate the extracted y_i segments. To improve the robustness of the counting model, we randomly insert segments unrelated to the target action between occurrences of the target action. Unrelated segments are the segments that are not extracted as occurrences of the target action in the weakly labeled segments from the target user. When we have other target actions (e.g., other workout types), we also use short segments of other target actions as unrelated segments.
3. To generate diverse training data, we also leverage data augmentation techniques mentioned in Section 3.3.1.

3.4.2. Fine-Tuning WeakCounter-Net

We fine-tune the pretrained WeakCounter-Net on the fine-tuning dataset generated in the previous procedure. The learning rate used in this procedure is smaller than that of the pretraining phase, as shown in **Table 1**.

4. EXPERIMENTS

4.1. Dataset and Evaluation Methodology

We collected 100 Hz three-axis acceleration data from ASUS Zen Watch three devices worn on the right wrists of nine subjects. A data collection session for an input segment contained three types of workouts in a random order: push-ups, squats, and sit-ups. The number of occurrences of each action was randomly

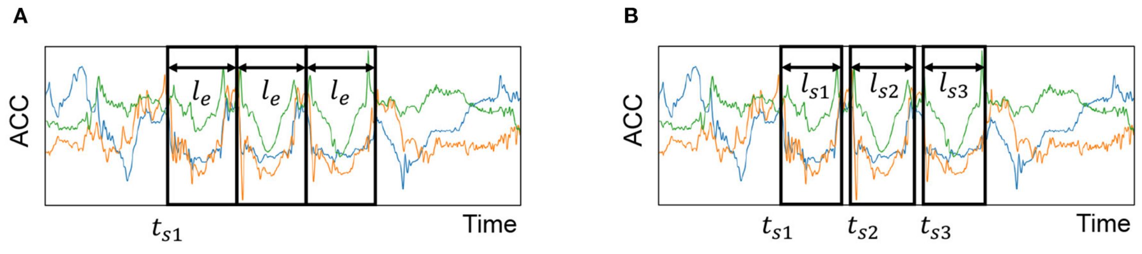

FIGURE 7 | Determining start time and length of each occurrence of a target action in a weakly labeled segment for sit-up workout. **(A)** Determining rough start time of the first occurrence t_{s1} and duration l_e. In this case, we assume that the duration of the 2nd and 3rd occurrences is also l_e. Therefore, the start times of the 2nd and 3rd occurrences are decided based on t_{s1} and l_e. **(B)** Determining start time and duration of each occurrence using optimization.

Algorithm 2 Determining the start time of the first occurrence and the length of target action.

```
Input: $ACC_p, y_t, \boldsymbol{L_{can}}$
Output: $l_e, t_{s1}$
1: /* find the optimal length $l_e$ in candidates $\boldsymbol{L_{can}}$ */
2: $\boldsymbol{DTW_{can}} = []$, $\boldsymbol{Seg_{can}} = []$
3: for $l_c$ in $\boldsymbol{L_{can}}$ do
4:   /* calculate Equation (1) by changing start time of the first occurrence */
5:   $\boldsymbol{D} = []$
6:   for $x = 0 \dots l - l_e$ do
7:     $\boldsymbol{Segs} = [s(x, l_c), \dots, s(x + (y_t - 1)l_c, l_c)]$ /* extracting short segments, with each of them corresponding to an occurrence */
8:     $\boldsymbol{P_{seg}} \leftarrow$ all pairs of $\boldsymbol{Segs}$
9:     $d_{sum} = 0$ /* storing sum of DTW over all pairs of $\boldsymbol{Segs}$ */
10:    for $p$ in $\boldsymbol{P_{seg}}$ do
11:      /* $p$ shows pair of two short segments and DTW($p$) calculates the distance between two segments */
12:      $d_{sum} += \mathrm{DTW}(p)$
13:    end for
14:    $\boldsymbol{D}$.append($d_{sum}$)
15:  end for
16:  /* scan local minima in $\boldsymbol{D}$ and remove cases where static segments are included in short segments */
17:  $\boldsymbol{V_m}, \boldsymbol{I_m} \leftarrow$ Value and index of each of detected minima in $\boldsymbol{D}$
18:  $STD_m = 0$ /* storing minimum standard deviation of the segment */
19:  /* repeat until $STD_m$ exceeds the threshold value to avoid including static segments */
20:  while $STD_m <= th_{std}$ do
21:    $v_{min} \leftarrow \min(\boldsymbol{V_m}), i_{min} \leftarrow \mathrm{argmin}(\boldsymbol{V_m})$
22:    $t_{s1} = i_{min}$
23:    $\boldsymbol{S} = [s(t_{s1}, l_c), \dots, s(t_{s1} + (y_t - 1)l_c, l_c)]$
24:    $STD_m = min([STD(s(t_{s1}, l_c)), \dots, STD(s(t_{s1} + (y_t - 1) * l_c, l_c))])$
25:    $\boldsymbol{V_m}$.pop($v_{min}$), $\boldsymbol{I_m}$.pop($i_{min}$)
26:  end while
27:  $d_{all} \leftarrow$ Sum of DTW over all pairs of $\boldsymbol{S}$
28:  $\boldsymbol{DTW_{can}}$.append($d_{all}$), $\boldsymbol{Seg_{can}}$.append($\boldsymbol{S}$)
29: end for
30: /* return $l_e$ and $t_{s1}$ that yield minimum Equation (1) */
31: $O_{dtw} \leftarrow \mathrm{argmin}(\boldsymbol{DTW_{can}})$
32: $l_e = \boldsymbol{L_{can}}[O_{dtw}]$
33: $t_{s1} \leftarrow$ Start point of $\boldsymbol{Seg_{can}}[O_{dtw}][0]$
```

selected between 1 and 20. Each session also contained unrelated motions such as drinking water and wiping the sweat. The subjects also obtained a few weakly labeled segments for fine-tuning by performing each action two to five times on different days. Specifically, we obtained four weakly labeled segments for fine-tuning for each subject. **Table 2** presents an overview of the dataset.

TABLE 1 | Parameters used in the methods and experiments.

Parameter	Value	Description
N_a	100	The number of segments to be generated from one segments in data augmentation
c_s	0.4	The magnitude for scaling in data augmentation
c_t	0.3	The magnitude for resampling with time-warping in data augmentation
c_b	5.0	The maximum absolute value to be shifted in data augmentation
n_s	4	The parameters to determine the length of the sub-segment in **Algorithm 1**
N_l	5	The number of candidates of the length of target action
w_s	100	The stride of sliding windows in **Algorithm 1**
N_f	100	The number of weakly labeled segments to be generated
lr_p	0.001	The learning rate for pretrainig
lr_f	0.0001	The learning rate for fine-tuning
e_p	100	The epoch number for pretraining
e_f	50	The epoch number for fine-tuning

Leave-one-subject-out cross-validation was performed during the experiment. Specifically, one subject was considered as a target user, and the remaining subjects were considered as source users. We generated 100 composite data from a few weakly labeled segments of a source user to train WeakCounter-Net. We also generated $5,000$ composite data from each weakly labeled segment of each target user for fine-tuning.

To validate the effectiveness of the proposed method, we prepared the following variants:

- WeakCounterF: This is the proposed method. This method fine-tunes a pretrained WeakCounter-Net on four weakly labeled segments from a target user (i.e., $y_t = 2, 3, 4, 5$).
- WeakCounterF (one-shot): This is the proposed method. This method fine-tunes a pretrained WeakCounter-Net on only one weakly labeled segment from a target user.
- CNN: This is a variant the proposed method. This method employs a 1D convolutional neural network instead of WeakCounter-Net. The network is composed of three 1D convolutional layers and one output layer.
- Only-composite: This method trains WeakCounter-Net on composite sensor data generated from weakly labeled data from a target user for fine-tuning. This method does not pretrain WeakCounter-Net on source users' data.
- WeakCounter: This is our prior method that does not use few-shot learning (Nishino et al., 2021). Therefore, this method does not fine-tune WeakCounter-Net on a target user's weakly labeled data.

We evaluated these variants based on the mean absolute error (MAE) (Skawinski et al., 2019) of their estimates. The experimental parameters used in our experiments are listed in **Table 1**.

TABLE 2 | Overview of dataset.

Subject	Number of segments	Average length of segment (s)	Number of weakly labeled segments	Average length of weakly labeled segments (s)	Age	Height (cm)	Weight (kg)
A	40	135.1	12	20.4	23	182	85
B	20	134.0	12	20.1	24	176	65
C	20	145.1	12	15.0	23	164	68
D	5	156.2	12	19.4	22	179	67
E	5	136.6	12	17.1	22	172	58
F	5	90.0	12	13.0	22	171	69
G	5	151.0	12	15.2	22	176	66
H	5	94.8.0	12	16.2	22	176	70
I	5	99.8	12	13.8	22	167	49

TABLE 3 | MAE of each method for each type of workout.

	Push-up	Squat	Sit-up	Average
CNN	1.02 ± 0.42	2.81 ± 2.05	1.02 ± 1.11	1.62 ± 1.57
Only-composite	3.32 ± 2.74	2.71 ± 1.89	1.64 ± 0.92	2.53 ± 2.04
WeakCounter	1.37 ± 0.89	2.03 ± 2.04	**0.88** ± 0.69	1.43 ± 1.38
WeakCounterF	**0.71** ± 0.65	1.35 ± 0.76	1.01 ± 0.60	**1.02** ± 0.70
WeakCounterF (one-shot)	0.93 ± 0.67	**1.32** ± 0.93	1.01 ± 0.71	1.09 ± 0.79

The bold value shows the lowest MAE for each type of workout.

TABLE 5 | MAE of each type of workout for WeakCounterF (one-shot) fine-tuned on a weakly labeled segment with label y_t.

y_t	Push-up	Squat	Sit-up	Average
2	0.86 ± 0.60	1.51 ± 0.92	0.99 ± 0.33	1.12 ± 0.70
3	0.84 ± 0.53	1.16 ± 0.96	1.00 ± 0.71	1.00 ± 0.74
4	1.13 ± 0.94	1.23 ± 0.83	1.01 ± 1.02	1.19 ± 0.90
5	0.84 ± 0.59	1.38 ± 1.14	0.83 ± 0.67	1.02 ± 0.84

TABLE 4 | MAE of each subject for each method.

	A	B	C	D	E
CNN	0.85 ± 0.10	**0.50** ± 0.28	2.80 ± 3.00	1.80 ± 1.91	1.73 ± 0.92
Only-composite	1.30 ± 0.31	4.85 ± 4.10	3.50 ± 2.08	2.60 ± 1.64	2.07 ± 2.48
WeakCounter	0.71 ± 0.25	0.65 ± 0.33	1.80 ± 1.26	3.13 ± 3.11	1.47 ± 0.31
WeakCounterF	**0.63** ± 0.32	0.67 ± 0.23	**1.43** ± 0.94	0.87 ± 1.03	1.20 ± 0.40
WeakCounterF (one-shot)	0.85 ± 0.47	0.71 ± 0.29	1.57 ± 0.90	**0.78** ± 0.71	**1.13** ± 0.51

	F	G	H	I	Average
CNN	1.27 ± 1.92	3.47 ± 1.33	1.00 ± 1.30	**1.13** ± 0.42	1.62 ± 1.57
Only-composite	1.80 ± 1.33	1.93 ± 1.78	2.13 ± 1.10	2.80 ± 2.09	2.55 ± 2.04
WeakCounter	**0.87** ± 0.46	1.73 ± 0.42	**0.53** ± 1.21	1.93 ± 1.70	1.43 ± 1.38
WeakCounterF	1.07 ± 0.31	**1.07** ± 0.12	0.60 ± 0.40	1.67 ± 1.47	**1.02** ± 0.70
WeakCounterF (one-shot)	1.10 ± 0.39	1.22 ± 0.62	0.55 ± 0.28	1.87 ± 1.39	1.09 ± 0.79

The bold value shows the lowest MAE for each subject.

4.2. Results

4.2.1. Performance

Table 3 lists the MAE of each method for each type of workout. As shown in the table, the proposed method achieved the lowest MAEs for these workout types. The average MAE for WeakCounterF is lower than that for WeakCounter by 0.41, indicating the effectiveness of fine-tuning WeakCounter-Net on few weakly labeled data. This table also highlights the effectiveness of one-shot learning because WeakCounterF (one-shot) also achieved lower MAEs than WeakCounter. The average MAE for CNN is greater than that for WeakCounterF by 0.60, indicating the important of the attention-based counting network. Additionally, the MAEs of Only-composite are much greater than those of the other methods. These results indicate that using only weakly labeled segments from a target user is ineffective. This is because it is difficult to train a robust counting network on weakly labeled segments only from a target user.

Table 4 lists the MAEs for each subject. In few cases, the MAEs for WeakCounterF are greater than those for WeakCounter, such as Subject E. However, WeakCounterF can reduce the MAEs in many cases. In the results of WeakCounter and CNN, the MAEs of few subjects are larger than 3.0 because of the large sensor data differences among the subjects. However, WeakCounterF can address these issues by employing composite data generated from weakly labeled segments from a target user. As shown in **Table 4**, CNN, Only-composite, and WeakCounter sometimes exhibit very poor performance for certain subjects. In contrast, WeakCounterF achieves stable performance for all the subjects.

4.2.2. Effect of the Number of Actions in Weakly Labeled Segments on One-Shot Learning

WeakCounterF (one-shot) employs a weakly labeled segment from a target user. Here, we investigate the relationship between counting performance and the number of occurrences of a target action (i.e., y_t) included in a weakly labeled segment. **Table 5** presents the results of WeakCounterF (one-shot) for different

TABLE 6 | MAE when detecting the start/end time of each occurrence for each y_t value (samples).

	y_t	Push-up	Squat	Sit-up	Average
Start points	2	25.35 ± 14.51	27.35 ± 31.80	50.09 ± 60.38	34.26 ± 40.36
	3	17.45 ± 13.44	28.42 ± 24.79	24.99 ± 24.03	23.62 ± 20.72
	4	18.68 ± 13.30	22.29 ± 12.80	31.47 ± 28.86	24.15 ± 19.78
	5	19.96 ± 13.44	19.12 ± 7.54	40.39 ± 21.74	26.49 ± 17.81
End points	2	21.71 ± 13.57	28.93 ± 31.20	56.76 ± 68.78	35.80 ± 45.27
	3	20.09 ± 16.83	25.03 ± 20.20	28.13 ± 24.47	24.42 ± 20.21
	4	16.60 ± 10.64	26.33 ± 14.81	38.02 ± 32.43	26.98 ± 22.48
	5	16.87 ± 18.51	17.78 ± 9.60	37.29 ± 23.78	23.98 ± 19.92
Average	2	23.53 ± 13.38	28.14 ± 31.10	53.42 ± 64.26	35.03 ± 42.46
	3	18.77 ± 14.46	26.73 ± 21.57	26.56 ± 24.00	24.02 ± 19.97
	4	17.64 ± 10.83	24.31 ± 13.23	34.74 ± 28.90	25.56 ± 19.96
	5	18.41 ± 12.88	18.45 ± 6.90	38.84 ± 21.80	25.23 ± 17.50

y_t values. We assumed that the occurrence extraction process is robust when y_t is high. Therefore, we assumed that the MAEs for larger y_t values will be smaller. However, as shown in **Table 5**, the MAEs for WeakCounterF (one-shot) were not significantly affected by the y_t values. This may be because WeakCounterF (one-shot) is robust against small errors in detecting each occurrence because this method adds random noises to the estimated start time and length (end time) of each occurrence when extracting the occurrences.

4.2.3. Error in Detecting Start and End Times of Each Occurrence

Here, we investigate the performance of our method for detecting the start and end times of each occurrence. **Table 6** lists the errors (in the samples) of the proposed method. The errors were calculated based on the manually annotated ground truth for the start and end times of each occurrence. Because the sampling rate of the sensor was 100 Hz, an error of 25 samples corresponds to 0.25 s. The results reveal that the MAEs when $y_t = 2$ are greater than those in the other cases. As mentioned above, when the number of occurrences included in a weakly labeled segment from a target user is large, the prediction of the duration of a target action is robust.

Here, the average duration of each occurrence of the push-up, squat, and sit-up actions were 177.8, 229.1, and 269.7 (samples), respectively. To improve the robustness, when we extract each occurrence, we randomly changed the start/end times by adding r to the predicted start/end times where $-0.1l_e < r < 0.1l_e$. A value of $0.1l_e$ corresponds to 10% of the estimated duration of the target action. Therefore, we consider that the error in the start/end time detection to be reasonable.

5. DISCUSSION

While the above results revealed the effectiveness of the proposed method, there is room for performance improvement. Although the errors in detecting the start/end time of each occurrence of a target action were small on average, we identified several cases where the proposed detection method yielded large errors depending on target actions and subjects, with the maximum error of 180 samples. Because the estimation of the duration of a target action did not have large errors, it is considered that the challenge lies in the estimation of the start time. We believe that the use of techniques such as motif detection can improve the estimation of the start/end time.

The detection results of the start/end time and the results of WeakCounterF (one-shot) for each y_t indicate that, even if the start/end times are accurately detected, the proposed method cannot always achieve high counting performance. When y_t is 4 in push-up, the start and end times were detected with smaller errors than the other y_t values. However, the counting error for $y_t = 4$ by WeakCounterF (one-shot) was larger than the other y_t. The results indicate that there are some issues regarding the method of generating composite data from extracted segments. As a part of our future work, we plan to generate more diverse training data by performing additional diversification operations.

In addition, because our experiment was conducted offline with well-segmented data, we plan to validate our method on real-time data. We believe that WeakCounterF also achieves good performance for real-time data by simply extracting segments with a certain length from the real-time data and feeding them into WeakCounter-Net. Note that, when we extract segments from the real-time data, it is important to avoid setting a breakpoint of the extraction at an occurrence of a target action. Similar to the data augmentation procedure, we can find moments with small variances and set the moments as the breakpoints.

6. CONCLUSION

This study proposed a new method for the few-shot weakly supervised repetition counting of human actions such as workouts using a wearable inertial sensor. The proposed method leverages few weakly labeled segments containing the occurrences of a target action from a target user to achieve precise repetition counting. Our experiments revealed that few-shot and one-shot learning based on our method achieved small errors for repetition counting. As a part of our future work, we plan to improve the proposed method to achieve few-shot weakly supervised real-time repetition counting.

DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article/**Supplementary Material**, further inquiries can be directed to the corresponding author.

ETHICS STATEMENT

Ethical review and approval was not required for the study on human participants in accordance with the local legislation and institutional requirements. Written informed consent for participation was not required for this study in accordance with the national legislation and the institutional requirements.

AUTHOR CONTRIBUTIONS

YN performed the method design, software implementation, data collection, data analysis, and article writing. TM conceived and directed the study, performed method design, data analysis, and article writing. TH performed method design and data analysis. All authors contributed to the article and approved the submitted version.

FUNDING

This work was partially supported by JSPS KAKENHI Grant Numbers JP21H03428, JP21H05299, and JP21K19769.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fcomp.2022.925108/full#supplementary-material

REFERENCES

Aehnelt, M., Gutzeit, E., Urban, B. (2014). "Using activity recognition for the tracking of assembly processes: challenges and requirements," in *Workshop on Sensor-Based Activity Recognition (WOAR)* (Rostock), 12–21.

Bian, S., Rey, V. F., Hevesi, P., and Lukowicz, P. (2019). "Passive capacitive based approach for full body gym workout recognition and counting," in *2019 IEEE International Conference on Pervasive Computing and Communications (PerCom)* (Kyoto), 1–10. doi: 10.1109/PERCOM.2019.8767393

Deng, S., Hua, W., Wang, B., Wang, G., and Zhou, X. (2020). "Few-shot human activity recognition on noisy wearable sensor data," in *International Conference on Database Systems for Advanced Applications* (Jeju: Springer), 54–72. doi: 10.1007/978-3-030-59416-9_4

Feng, S., and Duarte, M. F. (2019). Few-shot learning-based human activity recognition. *Expert Syst. Appl*. 138, 112782. doi: 10.1016/j.eswa.2019.06.070

Guo, X., Liu, J., and Chen, Y. (2017). "FitCoach: virtual fitness coach empowered by wearable mobile devices," in *IEEE Conference on Computer Communications (IEEE INFOCOM 2017)* (Atlanta, GA), 1–9. doi: 10.1109/INFOCOM.2017.8057208

Hu, D. H., Zheng, V. W., and Yang, Q. (2011). Cross-domain activity recognition *via* transfer learning. *Pervas. Mob. Comput*. 7, 344–358. doi: 10.1016/j.pmcj.2010.11.005

Kingma, D. P., and Ba, J. (2014). Adam: a method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. doi: 10.48550/arXiv.1412.6980

Lukowicz, P., Ward, J. A., Junker, H., Stäger, M., Tröster, G., Atrash, A., et al. (2004). "Recognizing workshop activity using body worn microphones and accelerometers," in *International Conference on Pervasive Computing* (Vienna), 18–32. doi: 10.1007/978-3-540-24646-6_2

Maekawa, T., Nakai, D., Ohara, K., and Namioka, Y. (2016). "Toward practical factory activity recognition: unsupervised understanding of repetitive assembly work in a factory," in *The 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (Heidelberg), 1088–1099. doi: 10.1145/2971648.2971721

Morales, J., Yoshimura, N., Xia, Q., Wada, A., Namioka, Y., and Maekawa, T. (2022). "Acceleration-based human activity recognition of packaging tasks using motif-guided attention networks," in *IEEE Int'l Conference on Pervasive Computing and Communications (PerCom 2022)* (Pisa), 1–12. doi: 10.1109/PerCom53586.2022.9762388

Morris, D., Saponas, T. S., Guillory, A., and Kelner, I. (2017). "Data augmentation of wearable sensor data for Parkinson's disease monitoring using convolutional neural networks," in *The 19th ACM International Conference on Multimodal Interaction* (New York, NY), 216–220.

Mortazavi, B. J., Pourhomayoun, M., Alsheikh, G., Alshurafa, N., Lee, S. I., and Sarrafzadeh, M. (2014). "Determining the single best axis for exercise repetition recognition and counting on smartwatches," in *11th International Conference on Wearable and Implantable Body Sensor Networks* (Zurich), 33–38. doi: 10.1109/BSN.2014.21

Nishino, Y., Maekawa, T., and Hara, T. (2021). "Weakcounter: acceleration-based repetition counting of actions with weakly supervised learning," in *2021 International Symposium on Wearable Computers* (Virtual), 144–146. doi: 10.1145/3460421.3480431

Pernek, I., Hummel, K. A., and Kokol, P. (2013). Exercise repetition detection for resistance training based on smartphones. *Pers. Ubiquit. Comput*. 17, 771–782. doi: 10.1007/s00779-012-0626-y

Seeger, C., Buchmann, A. P., and Van Laerhoven, K. (2011). "myhealthassistant: a phone-based body sensor network that captures the wearer's exercises throughout the day," in *BodyNets* (Beijing), 1–7. doi: 10.4108/icst.bodynets.2011.247015

Skawinski, K., Roca, F. M., Findling, R. D., and Sigg, S. (2014). "Recofit: using a wearable sensor to find, recognize, and count repetitive exercises," in *The SIGCHI Conference on Human Factors in Computing Systems* (Toronto, ON), 3225–3234.

Skawinski, K., Roca, F. M., Findling, R. D., and Sigg, S. (2019). "Workout type recognition and repetition counting with CNNs from 3D acceleration sensed on the chest," in *International Work-Conference on Artificial Neural Networks* (Gran Canaria), 347–359. doi: 10.1007/978-3-030-20521-8_29

Soro, A., Brunner, G., Tanner, S., and Wattenhofer, R. (2019). Recognition and repetition counting for complex physical exercises with deep learning. *Sensors* 19, 714. doi: 10.3390/s19030714

Sundholm, M., Cheng, J., Zhou, B., Sethi, A., and Lukowicz, P. (2014). "Smart-mat: recognizing and counting gym exercises with low-cost resistive pressure sensing matrix," in *The 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (Seattle, WA), 373–382. doi: 10.1145/2632048.2636088

van Kasteren, T. L. M., Englebienne, G., and Kröse, B. J. A. (2010). "Transferring knowledge of activity recognition across sensor networks," in *Pervasive Computing*, eds P. Floréen, A. Krüger and M. Spasojevic (Helsinki: Springer), 283–300. doi: 10.1007/978-3-642-12654-3_17

Wang, J., Zheng, V. W., Chen, Y., and Huang, M. (2018). "Deep transfer learning for cross-domain activity recognition," in *The 3rd International Conference on Crowd Science and Engineering* (Singapore), 1–8. doi: 10.1145/3265689.3265705

Xia, Q., Korpela, J., Namioka, Y., and Maekawa, T. (2020). Robust unsupervised factory activity recognition with body-worn accelerometer using temporal structure of multiple sensor data motifs. *Proc. ACM Interact. Mob. Wearab. Ubiquit. Technol.* 4, 1–30. doi: 10.1145/3411836

Xia, Q., Wada, A., Korpela, J., Maekawa, T., and Namioka, Y. (2019). Unsupervised factory activity recognition with wearable sensors using process instruction information. *Proc. ACM Interact. Mob. Wearab. Ubiquit. Technol.* 3, 1–23. doi: 10.1145/3328931

Xiao, T., Xu, Y., Yang, K., Zhang, J., Peng, Y., and Zhang, Z. (2015). "The application of two-level attention models in deep convolutional neural network for fine-grained image classification," in *The IEEE Conference on Computer Vision and Pattern Recognition* (Boston, MA), 842–850.

Yu, X., Vu, N. T., and Kuhn, J. (2019). "Learning the dyck language with attention-based Seq2Seq models," in *The 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP* (Florence), 138–146. doi: 10.18653/v1/W19-4815

Yu, X., Wu, X., Luo, C., and Ren, P. (2017). Deep learning in remote sensing scene classification: a data augmentation enhanced convolutional neural network framework. *GISci. Remote Sens.* 54, 741–758. doi: 10.1080/15481603.2017.1323377

Zeng, M., Gao, H., Yu, T., Mengshoel, O. J., Langseth, H., Lane, I., et al. (2018). "Understanding and improving recurrent networks for human activity recognition by continuous attention," in *The 2018 ACM International Symposium on Wearable Computers* (Singapore), 56–63. doi: 10.1145/3267242.3267286

Zhu, C., Byrd, R. H., Lu, P., and Nocedal, J. (1997). Algorithm 778: L-BFGS-B: fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.* 23, 550–560. doi: 10.1145/279232.279236

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.