Multi-Temporal Sampling Module for Real-Time Human Activity Recognition

JAEGYUN PARK, WON-SEON LIM, DAE-WON KIM, (Member, IEEE), AND JAESUNG LEE

1School of Computer Science and Engineering, Chung-Ang University, 221, Heukseok-Dong, Dongjak-Gu, Seoul 06974, Republic of Korea
2Department of Artificial Intelligence, Chung-Ang University, 221, Heukseok-Dong, Dongjak-Gu, Seoul 06974, Republic of Korea

Corresponding author: Dae-Won Kim (dwkim@cau.ac.kr) and Jaesung Lee (jslee.cau@gmail.com).

This research was partly supported by the Chung-Ang University research grant in 2021 and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2020R1A2C10135751).

ABSTRACT Human activity recognition, which recognizes human activities from time-series signals collected by sensors, is an important task in human-centered intelligent systems such as in healthcare and smart vehicles. In these applications, rapid response of the system is necessary because critical events such as an elderly person falling or drowsy driving require immediate action. A straightforward approach to achieving this requirement is to reduce the amount of information the model must handle. To this end, traditional studies have attempted to abstract the original signal by sampling it with a predefined interval. However, it is difficult to achieve the best efficiency because the ideal sampling interval is unknown in advance. In this study, we propose a multi-temporal sampling module that allows the neural networks to consider multiple sampling intervals simultaneously. Experiments on four benchmark datasets showed that the proposed model achieved the best F1 score over seven conventional models under the computation budget of 10M multiply-accumulate operations. Especially, an experiment on PAMAP2 dataset demonstrated that the proposed model can achieve the best trade-off between efficiency and accuracy when the input signal is oversampled at a high sampling frequency. In addition, the proposed model achieved ∼1,000× improvement with respect to model size compared to the conventional methods.

INDEX TERMS Activity recognition, real-time systems, neural networks, signal sampling.

I. INTRODUCTION

HUMAN Activity Recognition (HAR) aims to identify the daily activities of humans from time-series signals collected by sensors [1]. In most HAR studies, machine learning approaches have reported high performance [2]–[4] but require sophisticated feature engineering that incurs a lot of expert knowledge and time. For this reason, recent studies have increasingly considered Deep Learning (DL) models because they yield a superior learning performance based on how they automatically engineer features [5], [6]. As most DL models have high latency due to their complex architectures, 1D Convolutional Neural Networks (CNNs), which are supported by various hardware accelerators [7], are often considered to achieve real-time HAR [1], [6], [8].

In traditional HAR [9], [10], the original signal is sampled according to a pre-defined interval to improve efficiency, because the signal is typically oversampled [11]. Conventional CNNs for HAR, alternatively, can abstract the signal or feature map by pooling layers [12], [13]. However, it is difficult to achieve the best efficiency because the ideal sampling interval is unknown in advance. A straightforward solution to solve this issue is to allow CNNs to consider multiple sampling intervals simultaneously, yet the stride of pooling is fixed and the pooling is applied after the vanilla convolutions have already consumed a significant computational cost [14], [15].

To address this issue, we design a novel Multi-Temporal Sampling (MTS) module. Specifically, the module contains a new convolution unit, namely Sparse Sampling Convolution (S2Conv), that takes the random permuted form of a diagonal matrix to abstract the information delivered from the prior layer based on the sampling. Multiple S2Conv units that work with different sampling intervals are integrated into our MTS module to consider multiple sampling intervals simultaneously. By stacking MTS modules, the information to be handled can be reduced significantly as it passes through each layer. Our experiments demonstrate that our MTS-ConvNet achieves ∼1,000× and ∼15× improvements with respect to the conventional methods.
FIGURE 1: Design goal of a new component to replace a vanilla convolution (VanillaConv) unit. As temporal features pass through components, the amount of information to be handled is exponentially reduced, resulting in a reduction of computational cost and model size. (a): new feature maps are generated through VanillaConv from the original signals; (b): our components efficiently extract higher-level features by abstracting the information delivered from prior layers; (c): Based on the final features extracted by the components, an activity on each pattern is predicted. All types of convolutions come with rectified linear units.

II. RELATED WORK

In recent years, many HAR studies have proposed large DL models to solve various challenges derived from real-world applications. Some studies proposed DL models for complex activity recognition, including composite activities [16], [17], concurrent activities [18], [19]. To address multi-modal sensory datasets, some other studies integrated an attention mechanism into the DL models [20]–[24]. These models, including vanilla convolutions and recurrent units, demonstrated their effectiveness, but they are insufficient to ensure the real-time response when the available computation budget is reduced due to low-cost devices or background APPs [25]. Especially, the recurrent units often require infeasible computational costs at edge implementation [7], [26], thus making the real-time HAR difficult.

In HAR literature, many comprehensive surveys have emphasized the importance of real-time applications [1], [6], [8], [27], [28]. Most prior studies have focused on sampling the signals and experimentally demonstrating that the ideal sampling interval depends on the type of activity [9], [11], [29]. Furthermore, Cheng et al. attempted to predict an instance-wise sampling interval [10]. Similarly, Yang et al. proposed an instance-wise dynamic sensor selection method [30]. However, these approaches, which focus on data acquisition, may result in irretrievable information loss.

Some early studies proposed DL models for real-time HAR, including pre-processing that transforms the input data from the time domain to the frequency domain [31], [32]. However, this complex pre-processing should be conducted without stopping, resulting in an increase in overhead. Ignatov first attempted to design a CNN architecture for real-time HAR without any transformation of the data [12]. Wan et al. proposed a CNN architecture including three convolutional layers for real-time HAR using accelerometers and gyroscopes [13]. However, these studies have not examined the effects of the sampling interval on performance for HAR.

III. MULTI-TEMPORAL SAMPLING MODULE

Our goal is to design a novel component to replace a vanilla convolution (VanillaConv) unit for real-time HAR. The component abstracts the information delivered from the prior layer, resulting in an improvement of efficiency. Figure 1 shows the forward pass of temporal information in the proposed neural network. Given original signals, a VanillaConv unit generates new temporal feature maps by learning intra-modality information or inter-modality correlations [8]. As the components are stacked along with the layer, the information to be handled is exponentially reduced. Because the ideal sampling interval may also vary when the information is abstracted, the component must consider multiple sampling intervals at each layer. To this end, we will design an integrated module as the basic component. Finally, a fully-connected (FC) layer predicts a human activity using extracted features.

A. VANILLA CONVOLUTION AND NOTATIONS

Let $I \in \mathbb{Z}^{T,M}$ and $O \in \mathbb{Z}^{T,N}$ be an input and output for VanillaConv, respectively, where $T$ is the temporal resolu-
In Eq. (3), the feature map sampling is applied to time-wise convolution, similarly to [34]. Given the kernel size \( K \), the feature map-wise convolution, the \( m \)-th feature map at time \( t \) is calculated as

\[
\hat{O}_{t,m} = \sum_{i} K_{i,m} I_{t+i,m}
\]

Next, for a given kernel \( K^2 \in \mathbb{Z}^{M,N} \) of time-wise convolution, the \( n \)-th feature map is calculated as

\[
O_{t,n} = \sum_{m} K_{m,n} \hat{O}_{t,m}
\]

where the computational cost is \( T \times M \times (D_k + N) \). Based on this decomposition, we integrate two sampling processes into VanillaConv, i.e., time sampling and feature map sampling, as shown in Figure 2.

**B. SPARSE SAMPLING CONVOLUTION**

We design an efficient convolution unit that extracts higher-level features at a specific sampling interval within MTS modules. In Eq. (2), the feature map sampling is applied to \( \hat{O} \). Let \( G^n \in \mathbb{Z}^{T,M} \) be the sampled feature maps used to generate the \( n \)-th output feature map \( O_{n,m} \), where \( M_s \ll M \) is the number of the sampled feature maps. As a result, the size of the kernel \( K^2 \) will be reduced, i.e., \( K^2 \in \mathbb{Z}^{M,N} \). Meanwhile, the time sampling is applied to \( K^1 \) in Eq. (1).

We consider the kernel size \( D_k \) as the sampling interval and thus parameters of \( K^1 \) are sampled with \( D_k \). The sampled kernel \( \tilde{K}^1 \) can be written as

\[
\tilde{K}^1_{i,m} = \begin{cases} 
\theta, & \text{if } i = i_m \\
0, & \text{otherwise}
\end{cases}
\]

where \( \theta \) is a trainable parameter and \( i_m \) is feature map-dependent index that has only one value in \( \tilde{K}^1_{i,m} \in \mathbb{Z}^{D_k} \). To extract temporal information, \( O_{t,n} \) should be computed by interacting with features of the adjacent times. To this end, the time sampling can be conducted by spreading \( i_m \) of Eq. (4) across adjacent times as shown in Figure 2(c), while satisfying the following condition:

\[
\forall i : \sum_{m} \tilde{K}^1_{i,m} \neq 0
\]

Furthermore, the feature map sampling can be directly applied to \( I \) because the order of input feature maps is unchanged by \( \tilde{K}^1 \). Additionally, the parameters of \( \tilde{K}^1 \) can be replaced with one by the associative law of multiplication between them and the parameters of \( K^2 \) of Eq. (3). As a result, two kernels can be integrated into \( \tilde{K} \in \mathbb{Z}^{M_s,N} \) without additional computation.

Meanwhile, it is difficult to find the optimal \( M_s \) with respect to a trade-off between accuracy and efficiency. Therefore, we simply set \( M_s \) to \( D_k \), which is the minimum number of parameters within \( \tilde{K} \) that satisfy Eq. (5). Consequently, we propose a Sparse Sampling Convolution (S2Conv) of which the computational cost is \( T \times D_k \times N \). It extracts the \( n \)-th output feature map at time \( t \) as follow:

---

**FIGURE 2**: Illustration of key elements of our Multi-Temporal Sampling (MTS) module. (a): The MTS module includes Sparse Sampling Convolution (S2Conv) units with multiple sampling intervals, where a kernel size \( D_k \) is considered as a sampling interval; (b): some input feature maps are sampled and used to generate each output feature map, wherein a time-wise convolution learns about the group mapping; (c): our S2Conv efficiently extracts an output feature map per group via time sampling.
Algorithm 1 Multi-Temporal Sampling module

1: Input: \( I \in \mathbb{Z}^{T,M}, S^k \); \( \triangleright \) the set of the kernel sizes \( S^k \)
2: Output: \( O \in \mathbb{Z}^{T,N} \)
3: \( N_k \leftarrow \left\lfloor \frac{N}{|S^k|+1} \right\rfloor \) \( \triangleright \) # of output channels for branches
4: for \( n = 1 \) to \( N_k \) do
5: for \( t = 1 \) to \( T \) do
6: \( \tilde{I}_{t,n} \leftarrow \sum_{m} K_{m,n} I_{t,m} \) \( \triangleright \) TWConv
7: end for
8: end for
9: \( I \leftarrow \text{ReLU}(\text{BN}(\tilde{I})) \) \( \triangleright \) the batch normalization BN
10: \( b \leftarrow 0 \)
11: for \( D_k \in S^k \) do \( \triangleright \) the user-defined parameter
12: \( b \leftarrow b + 1 \)
13: \( \hat{O}^b \leftarrow \text{calculate S2Conv}(\tilde{I}, D_k) \) \( \triangleright \) use Eq. (6)
14: \( \hat{O}^b \leftarrow \text{ReLU}(\text{BN}(\hat{O}^b)) \) \( \triangleright \hat{O}^b \in \mathbb{Z}^{T,N_k} \)
15: end for
16: \( O \leftarrow \text{Concat}(\tilde{I}, \hat{O}) \) \( \triangleright \) concatenate them at channel axis

\[
O_{t,n} = \sum_{i} K_{i,n} G^m_{t+i,n} \tag{6}
\]

C. PERMUTATION OF FEATURE MAPS

To avoid loss of information resulting from the random sampling of feature maps, the following issues should be considered. First, mapping \( M \) input feature maps to \( N \) sample groups generates a large search space, resulting in \( MCD_k \) possible cases per group. Second, the order of sampled feature maps within \( G^m \) should be considered because their dependency can vary across time. To handle these issues, we permute the input feature maps and map them to each sample group in order. Let \( \mathcal{P}_1 \) and \( \mathcal{P}_2 \) be permutation functions for grouping and an arrangement of feature maps within each group, respectively. Based on this, Eq. (6) can be rewritten as follows:

\[
O = \mathcal{K} \circ \mathcal{P}_2 \circ \mathcal{P}_1 (I)
\]

\[
= \mathcal{K}(IP) \tag{7}
\]

where \( \circ \) is a function composition and \( \mathcal{K} \) is computed by Eq. (6). Because \( \mathcal{P}_1 \) and \( \mathcal{P}_2 \) can be computed through the permutation matrix, the composition of them can be conducted by multiplying a permutation matrix \( P \in \mathbb{Z}^{M,M} \) to \( I \in \mathbb{Z}^{T,M} \).

However, it is difficult for \( P \) to be optimized by stochastic gradient descent because it has discrete values. This issue can be resolved by training a permuted time-wise convolution directly [35]. Specifically, the input feature maps to each module are permuted by time-wise convolution. After that, feature maps can be simply sampled in sequence. Given a permuted input \( \tilde{I} \in \mathbb{Z}^{T,M} \), when \( M = N \), each feature map group \( G^m \) is sampled as follows:

\[
G^m = \tilde{I}_{:,n-[D_k/2]:n+|D_k/2]} \tag{8}
\]

FIGURE 3: The neural architecture of our MTS-ConvNet. All types of convolutions are represented by the form of “convolution (kernel size), the number of output feature maps.”

The number of feature maps for the permuted input is controlled by the time-wise convolution within the MTS module. As a result, according to Eq. (6) and Eq. (8), our S2Conv unit substantially reduces the computational cost and model size compared with the VanillaConv via both the time sampling and feature map sampling.

D. CONSTRUCTING MODULE AND ARCHITECTURE

Built on the S2Conv units with multiple sampling intervals or kernel sizes \( D_k \), we propose a Multi-Temporal Sampling (MTS) module. Algorithm 1 represents the pseudocode for the forward pass of our MTS module. Given the input feature maps and the set of the kernel sizes, its branches extract the same number of output feature maps (Line 3), where the set of the kernel sizes is the user-defined parameter. In Lines 4–7, the input feature maps are then permuted by the Time-Wise
TABLE 1: Comparison results from four datasets. \(\nabla\)/\(\triangle\) indicates that the corresponding model is significantly worse/better than our MTS-ConvNet based on paired \(t\)-test at 95% significance level on three metrics.

| Model            | Objective      | OPPOORTUNITY | WISDM | MACs | Model size | F1 score |
|------------------|----------------|--------------|-------|------|------------|----------|
|                  |                |               |       | MACs |            |          |
| MTS-ConvNet (Ours) | Real-time HAR | 2.381M \(\nabla\) | 0.849M | 5.00M | 0.18M | 0.864 |
| Ignatov [12]     | Real-time HAR  | 7.895M \(\nabla\) | 6.490M \(\nabla\) | 0.925\(\nabla\) | 3.685M \(\nabla\) | 3.068M \(\nabla\) | 0.840\(\nabla\) |
| Wan et al. [13]  | Time-series classification | 2.199M \(\nabla\) | 0.228M \(\nabla\) | 0.939 | 0.968M \(\nabla\) | 0.161M \(\nabla\) | 0.862 |
| ResNet           | Time-series classification | 61.769M \(\nabla\) | 0.483M \(\nabla\) | 0.952\(\nabla\) | 28.862M \(\nabla\) | 0.481M \(\nabla\) | 0.893\(\nabla\) |
| FCN              | Time-series classification | 34.440M \(\nabla\) | 0.269M \(\nabla\) | 0.958 | 15.983M \(\nabla\) | 0.267M \(\nabla\) | 0.880\(\nabla\) |
| ResNext          | Efficient design for image classification | 27.466M \(\nabla\) | 22.056M \(\nabla\) | 0.941 | 12.878M \(\nabla\) | 22.035M \(\nabla\) | 0.869 |
| MobileNetV2      | Efficient design for image classification | 11.997M \(\nabla\) | 2.189M \(\nabla\) | 0.925\(\nabla\) | 5.972M \(\nabla\) | 2.188M \(\nabla\) | 0.861 |
| SqueezeNet       | Efficient design for image classification | 5.338M \(\nabla\) | 0.360M \(\nabla\) | 0.920\(\nabla\) | 2.146M \(\nabla\) | 0.358M \(\nabla\) | 0.845\(\nabla\) |

Convolution (TWConv). The permuted feature maps are fed into branches of MTS module with different \(D_k\) (Line 11–15). In Line 16, their outputs and the permuted input are concatenated. All convolutions come with a Batch Normalization (BN) and Rectified Linear Unit (ReLU); herein, the BN layers include biases.

In the MTS module, the time complexity is determined by computations of TWConv and S2Conv. By assuming that elements of \(S^k\) are an arithmetic sequence of which the first term and the common difference are three and two, respectively, their computations cost can be computed as follows:

\[
(M + |S^k|(|S^k| + 2)) \frac{N}{|S^k| + 1} T
\]

Therefore, the time complexity of the MTS module is \(O(|S^k| (M + \frac{M}{|S^k|}) NT)\). Suppose that VanillaConv’s \(D_k\) is a mean of \(S^k\). Then, its time complexity can be computed as \(O(|S^k| M N T)\). Because \(M \gg |S^k| \geq 2\) generally, \(|S^k| M > |S^k| + \frac{M}{|S^k|}\). Thus, our MTS module has lower time complexity than does VanillaConv.

Built on the MTS module, we constructed our MTS-ConvNet architecture as shown in Figure 3. For brevity, we represent all type of convolution with a specific kernel size as “convolution (kernel size)”. We used VanillaConv (1) to examine the temporal modeling capability of the MTS module and to improve the efficiency of MTS-ConvNet. Because a delayed reduction of resolution may lead to higher classification accuracy [36], the pooling layers were concentrated toward the end of the network. MTS-ConvNet for all datasets are trained in PyTorch [37] by Adam optimizer [38] with \(\beta_1 = 0.9, \beta_2 = 0.999\) and \(\epsilon = 10^{-8}\). Both the learning rate and weight decay are set as 0.0005. We train MTS-ConvNet for 500 epochs with a batch size of 128 on a 2080Ti Graphics Processing Unit (GPU).

IV. EXPERIMENTAL RESULTS

We conducted experiments using four benchmark datasets. The UC-I-HAR [39] and WISDM [40] datasets include six activities such as “jogging” and “walking”. They were collected at 50 and 20 Hz frequencies, and their segment lengths...
FIGURE 5: Comparison of feature maps extracted at different kernel sizes $D_k$ for two activities, “Ascending stairs” and “Standing.” Among the outputs of each operation, a feature map with the highest mean values across the time axis are shown.

are 128 and 60, respectively. Also, the UCI-HAR dataset was collected from accelerometer and gyroscope, while the WISDM dataset was collected from the only accelerometer. The OPPORTUNITY [41] and PAMAP2 [42] datasets contain 18 classes, including complex activities such as “playing soccer”. For real-time HAR, we only considered the on-body sensors. They were collected at 30 and 100 Hz, and their segment lengths are 150 and 512, respectively.

To evaluate the performance of our MTS-ConvNet for HAR, we used three metrics: Multiply-Accumulate (MAC) operation, model size, and F1 score. The MAC operation is the basic computation of neural networks, which takes the form $wx + b$. Therefore, we used MACs to measure the computations of the neural networks. Model size is typically used to measure the number of parameters in neural networks. Lastly, because HAR datasets often involve class imbalance issues [8], the F1 score is commonly used as an alternative to accuracy.

We adopted seven baseline networks, as follows.

- **Real-time HAR models.** Ignatov [12] introduced a CNN for real-time HAR that consists of a single convolutional and FC layer. In addition, Wan et al. [13] proposed a CNN that consists of three convolutional and two FC layers. Their experiments support baseline
time-series classification (TSC) models. We adopted two CNNs that have reported significant success for the TSC problem. The 1D Residual Neural Network (ResNet) [43] and Fully Convolutional Network (FCN) [44] include eleven and three convolutional layers, respectively.

- Efficient CNNs. We also use three CNNs: MobileNetV2 [45], SqueezeNet [36], and ResNext [46]. These were designed to produce a computationally efficient model for image classification. To conduct experiments on HAR datasets, we replaced 2D operations with 1D operations.

Table 1 shows the overall comparison results of our MTS-ConvNet along with all the baseline networks. We obtained the average value by repeating the experiment 10 times. Our MTS-ConvNet exhibited comparable F1 scores while attaining the lowest MACs and model size. Specifically, MTS-ConvNet achieved higher F1 scores than existing real-time HAR models on the UCI-HAR, WISDM, and PAMAP2 datasets. Furthermore, MTS-ConvNets trained on the OPPORTUNITY and PAMAP2 datasets had smaller model sizes than real-time HAR models trained on the UCI-HAR and WISDM datasets, which makes real-time HAR on complex activities possible.

Meanwhile, ResNet and FCN with the highest MACs achieved the best F1-score on UCI-HAR, WISDM, and OPPORTUNITY datasets. These results indicate that a deeper network and more output channels can produce better F1 scores. However, ResNet and FCN, which require at least 23 × MACs compared to MTS-ConvNet, can be insufficient to ensure the real-time response on wearable devices with various hardware specifications. On the other hand, our MTS-ConvNet outperformed ResNet and FCN in terms of the F1 score on PAMAP2 collected at 100 Hz even though 49 × and 29 × improvements with respect to MACs, respectively. This result indicates that the proposed method can achieve the best trade-off between efficiency and accuracy when the input signal is oversampled at a high sampling frequency.

V. ANALYSIS ON MODEL SIZE AND SAMPLING INTERVALS

Our MTS-ConvNet demonstrated substantial efficiency with respect to model size. Figure 4 shows a comparison result of the MTS-ConvNet and [13], which is a real-time HAR model built on VanillaConv using the UCI-HAR dataset. The two horizontal axes indicate a scale factor to control the number of output channels and the depth of VanillaConv units or MTS modules in each model, respectively. As shown in Figure 4, MTS-ConvNet has an efficient rate of increase in model size as the network becomes wider or deeper.

To demonstrate the benefit of considering multiple sampling intervals, we compared feature maps extracted at different kernel size $D_k$ as information for two activities passes through our MTS modules. For better visibility, we used the WISDM dataset, which was collected from only an accelerometer. As shown in Figure 5, the activity “Ascending stairs” has higher variations in input signals while “Standing” hardly has variations. For “Standing,” the temporal features tended to be emphasized from all S2Conv units, regardless
TABLE 2: Improvement of F1 score via hyper-parameter optimization of the MTS module.

|                  | UCI-HAR | WISDM |
|------------------|---------|-------|
| \(D_k\), Depth   | [3, 5, 7], 4 | [3.5, 7.9, 11, 13], 5 |
| MACs             | 1.089M \(\rightarrow\) 0.917M, 0.500M \(\rightarrow\) 0.427M |
| Model size       | 0.019M \(\rightarrow\) 0.017M, 0.018M \(\rightarrow\) 0.015M |
| F1 score          | 0.940 \(\rightarrow\) 0.947, 0.864 \(\rightarrow\) 0.876 |

**OPPORTUNITY**

|                  | PAMAP2 |
|------------------|--------|
| \(D_k\), Depth   | [3, 5], 3 |
| MACs             | 2.301M \(\rightarrow\) 2.560M, 5.174M \(\rightarrow\) 5.063M |
| Model size       | 0.029M \(\rightarrow\) 0.034M, 0.023M \(\rightarrow\) 0.022M |
| F1 score          | 0.849 \(\rightarrow\) 0.872, 0.754 \(\rightarrow\) 0.767 |

of the specific \(D_k\). Alternatively, for “Ascending stairs,” the feature maps extracted by S2Conv units with different \(D_k\) tended to have different waveforms (i.e., different abstracted information).

When a VanillaConv is replaced with an S2Conv, the amount of skipped information increases as \(D_k\) becomes higher, resulting in better efficiency while accuracy may be degraded. If the MTS module includes only a single S2Conv (3), the efficiency gain is reduced, while if it includes only a single S2Conv (7), the accuracy can be degraded. However, the ideal \(D_k\) is unknown in advance as shown in Figure 5.

Consequently, our MTS module can achieve a better trade-off between the efficiency gain and the accuracy by using multiple S2Conv units with diverse \(D_k\).

Furthermore, we examined the impact of the number of kernel sizes \(|D_k|\) along with a depth of network on the F1 score. As shown in Figure 6, MTS-ConvNet, which either uses more multiple kernel sizes or is deeper, tends to achieve a higher F1 score. Meanwhile, we found that the small number of filters in the first TWCong led to a poor F1 score on the OPPORTUNITY dataset, which was collected in a sensor-rich environment. This may be because 113 input channels are immediately compressed to 16 channels, resulting in a loss of information. Therefore, we modified the number of output channels of the first MTS module to 32 only for the OPPORTUNITY dataset. Finally, Table 2 shows the improved F1 scores.

VI. REAL-TIME ACTIVITY RECOGNITION

To estimate the actual response time of our MTS-ConvNet, we used a Samsung Galaxy S10 smartphone having an octa-core processor (\(2 \times 2.7\) GHz + \(2 \times 2.3\) GHz + \(4 \times 1.9\) GHz) and 8GB RAM; herein, our implementation excluded the use of graphics processing units because the real-time HAR should be conducted without stopping as background APPs. Specifically, input signals are acquired in real-time from the smartphone’s built-in accelerometer and gyroscope at 50 Hz. After that, our MTS-ConvNet, which was pretrained on the UCI-HAR dataset and embedded into the smartphone, continuously runs the activity recognition whenever the previous prediction is finished. As a result, a short YouTube demo video is available at [https://youtu.be/Ie47soUp6Bs](https://youtu.be/Ie47soUp6Bs).

Inference time of our MTS-ConvNet was measured between 20 and 45 ms. Because the length of recognition intervals for the UCI-HAR dataset is 128 (2.56s segments), our model is sufficient to meet the real-time requirement.

VII. CONCLUSIONS

In this paper, we present a novel approach to integrating a traditional sampling process into a neural network for real-time human activity recognition. To this end, we introduce a sparse sampling convolution unit that allows neural networks to abstract the information delivered from the prior layer based on the sampling, resulting in improved efficiency. Furthermore, we propose a novel multi-temporal sampling module that contains the proposed convolution units with multiple kernel sizes, resulting in a better trade-off between the efficiency gain and the accuracy. The proposed module enables a sophisticated architecture that depends on various resource-limited sensor devices. Therefore, promising study in the future would be a neural architecture search.

REFERENCES

[1] L. M. Dang, K. Min, H. Wang, M. J. Piran, C. H. Lee, and H. Moon, “Sensor-based and vision-based human activity recognition: A comprehensive survey,” Pattern Recognition, vol. 108, p. 107561, 2020.

[2] X. Yang, S. A. Shah, A. Ren, N. Zhao, Z. Zhang, D. Fan, J. Zhao, W. Wang, and M. Ur-Rehman, “Freezing of gait detection considering leaky wave cable,” IEEE Transactions on Antennas and Propagation, vol. 67, no. 1, pp. 554–561, 2018.

[3] X. Yang, L. Guan, Y. Li, W. Wang, Q. Zhang, M. U. Rehman, and Q. H. Abbasi, “Contactless finger tapping detection at c-band,” IEEE Sensors Journal, vol. 21, no. 4, pp. 5249–5258, 2020.

[4] H. Bi, M. Perello-Nieto, R. Santos-Rodriguez, and P. Flach, “Human activity recognition based on dynamic active learning,” IEEE Journal of Biomedical and Health Informatics, vol. 25, no. 4, pp. 922–934, 2020.

[5] J. Huang, S. Lin, N. Wang, G. Dai, Y. Xie, and J. Zhou, “Tsc-cnn: A two-stage end-to-end cnn for human activity recognition,” IEEE journal of biomedical and health informatics, vol. 24, no. 1, pp. 292–299, 2019.

[6] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu, “Deep learning for sensor-based activity recognition: A survey,” Pattern Recognition Letters, vol. 119, pp. 3–11, 2019.

[7] T. Zebin, P. J. Scully, N. Peek, A. J. Casson, and K. B. Ozanyan, “Design and implementation of a convolutional neural network on an edge computing smartphone for human activity recognition,” IEEE Access, vol. 7, pp. 133 509–133 520, 2019.

[8] K. Chen, D. Zhang, L. Yao, B. Guo, Z. Yu, and Y. Liu, “Deep learning for sensor-based human activity recognition: Overview, challenges, and opportunities,” ACM Computing Surveys (CSUR), vol. 54, no. 4, pp. 1–40, 2021.

[9] A. Khan, N. Hammerla, S. Mellor, and T. Plötz, “Optimising sampling rates for accelerometer-based human activity recognition,” Pattern Recognition Letters, vol. 73, pp. 33–40, 2016.

[10] W. Cheng, S. Erfani, R. Zhang, and R. Kotagiri, “Learning datum-wise sampling frequency for energy-efficient human activity recognition,” in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, no. 1, 2018.

[11] L. Zheng, D. Wu, X. Ruan, S. Weng, A. Peng, B. Tang, H. Lu, H. Shi, and H. Zheng, “A novel energy-efficient approach for human activity recognition,” Sensors, vol. 17, no. 9, p. 2064, 2017.

[12] A. Ignatov, “Real-time human activity recognition from accelerometer data using convolutional neural networks,” Applied Soft Computing, vol. 62, pp. 915–922, 2018.

[13] S. Wan, L. Qi, X. Xu, C. Tong, and Z. Gu, “Deep learning models for real-time human activity recognition with smartphones,” Mobile Networks and Applications, vol. 25, no. 2, pp. 743–755, 2020.
[14] M. Xu, F. Qian, M. Zhu, F. Huang, S. Pushp, and X. Liu, “Deephwar: Adaptive local offloading for on-wearable deep learning,” IEEE Transactions on Mobile Computing, vol. 19, no. 2, pp. 314–330, 2019.

[15] W. Chen, D. Yeh, Y. Zhang, and S. Pu, “All you need is a few shifts: Designing efficient convolutional neural networks for image classification,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 7241–7250.

[16] L. Peng, L. Chen, Z. Ye, and Y. Zhang, “Aroma: A deep multi-task learning based simple and complex human activity recognition method using wearable sensors,” Procedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, vol. 2, no. 2, pp. 1–16, 2018.

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

[18] T. Okita and S. Inoue, “Recognition of multiple overlapping activities using compositional cnn-lstm model,” in Procedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Procedings of the 2017 ACM International Symposium on Wearable Computers, 2017, pp. 165–168.

[19] Y. Zhang, X. Li, J. Zhang, S. Chen, M. Zhou, R. A. Farneth, I. Marsic, and R. S. Burd, “Car-a deep learning structure for concurrent activity recognition,” in 2017 16th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN). IEEE, 2017, pp. 299–309.

[20] V. S. Murahari and T. Plötz, “On attention models for human activity recognition,” in Procedings of the 2018 ACM international symposium on wearable computers, 2018, pp. 100–103.

[21] K. Chen, L. Yao, X. Wang, D. Zhang, T. Gu, Z. Yu, and Z. Yang, “Interpretable parallel recurrent neural networks with convolutional attention for multi-modality activity modeling,” in 2018 International Joint Conference on Neural Networks (IJCNN). IEEE, 2018, pp. 1–8.

[22] H. Ma, W. Li, X. Zhang, S. Gao, and S. Lu, “Attensense: Multi-level attention mechanism for multimodal human activity recognition,” in IJCAI, 2019, pp. 3019–3109.

[23] K. Chen, L. Yao, D. Zhang, X. Wang, X. Chang, and F. Nie, “A semi-supervised recurrent convolutional attention model for human activity recognition,” IEEE transactions on neural networks and learning systems, vol. 31, no. 5, pp. 1747–1756, 2019.

[24] K. Chen, L. Yao, D. Zhang, B. Gao, and Z. Yu, “Multi-agent attentional activity recognition,” in Procedings of the 28th International Joint Conference on Artificial Intelligence, 2019, pp. 1344–1350.

[25] X. Cheng, L. Zhang, Y. Tang, Y. Liu, H. Wu, and J. He, “Real-time human activity recognition using conditionally parametrized convolutions on mobile and wearable devices,” IEEE Sensors Journal, vol. 22, no. 6, pp. 5889–5901, 2022.

[26] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, “Convolutional sequence to sequence learning,” in International Conference on Machine Learning. PMLR, 2017, pp. 1243–1252.

[27] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, and P. J. Havinga, “A survey of online activity recognition using mobile phones,” Sensors, vol. 15, no. 1, pp. 2095–2085, 2015.

[28] J. Qi, B. Yang, A. Warach, Z. Deng, Y. Zhao, and Y. Yang, “Examining sensor-based physical activity recognition and monitoring for healthcare using internet of things: A systematic review,” Journal of biomedical informatics, vol. 87, pp. 138–153, 2018.

[29] J. Lee and J. Kim, “Energy-efficient real-time human activity recognition on smart mobile devices,” Mobile Information Systems, vol. 2016, 2016.

[30] X. Yang, Y. Chen, H. Yu, Y. Zhang, W. Lu, and R. Sun, “Instance-wise dynamic sensor selection for human activity recognition,” in Procedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 01, 2020, pp. 1104–1111.

[31] D. Ravi, C. Wong, B. Lo, and G.-Z. Yang, “A deep learning approach to on-node sensor data analytics for mobile or wearable devices,” IEEE journal of biomedical and health informatics, vol. 21, no. 1, pp. 56–64, 2016.

[32] G. Bhat, R. Deb, V. V. Chaurasia, H. Shill, and U. Y. Ogras, “Online human activity recognition using low-power wearable devices,” in 2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). IEEE, 2018, pp. 1–8.

[33] K. Fukushima and S. Miyake, “Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition,” in Competition and Cooperation in Neural Nets. Springer, 1982, pp. 267–285.

[34] L. Sifre, “Rigid-motion scattering for image classification,” Ph.D. dissertation, 2014.

[35] B. Wu, A. Wan, X. Yue, P. Jin, S. Zhao, N. Golmant, A. Gholaminejad, J. Gonzalez, and K. Keutzer, “Shift: A zero flop, zero parameter alterna-tive to spatial convolutions,” in Procedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 9127–9135.

[36] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, “SqueezeNet: Alexnet-level accuracy with 50x fewer parameters and &< 0.5 mb model size,” arXiv preprint arXiv:1602.07360, 2016.

[37] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga et al., “Pytorch: An imperative style, high-performance deep learning library,” Advances in neural information processing systems, vol. 32, pp. 8026–8037, 2019.

[38] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” arXiv preprint arXiv:1412.6980, 2014.

[39] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. Reyes-Ortiz, “A public domain dataset for human activity recognition using smartphones,” in 21th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN). CIACO, 2013, pp. 437–442.

[40] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, “Activity recognition using cell phone accelerometers,” ACM SigKDD Explorations Newsletter, vol. 12, no. 2, pp. 74–82, 2011.

[41] R. Chavarriaga, H. Sagha, A. Calatroni, S. T. Digumarti, G. Tröster, J. d. R. Millán, and D. Roggen, “The opportunity challenge: A benchmark database for on-body sensor-based activity recognition,” Pattern Recognition Letters, vol. 34, no. 15, pp. 2033–2042, 2013.

[42] A. Reiss and D. Stricker, “Introducing a new benchmarked dataset for activity monitoring,” in 2012 16th International Symposium on Wearable Computers. IEEE, 2012, pp. 108–109.

[43] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, “Deep learning for time series classification: a review,” Data Mining and Knowledge Discovery, vol. 33, no. 4, pp. 917–963, 2019.

[44] Z. Wang, W. Yan, and T. Oates, “Time series classification from scratch with deep neural networks: A strong baseline,” in 2017 International joint conference on neural networks (IJCNN). IEEE, 2017, pp. 1578–1585.

[45] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks,” in Procedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 4510–4520.

[46] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks,” in Procedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1492–1500.