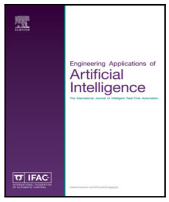
Engineering Applications of Artificial Intelligence 124 (2023) 106543

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



GTSNet: Flexible architecture under budget constraint for real-time human activity recognition from wearable sensor

Jaegyun Park [a], Won-Seon Lim [a], Dae-Won Kim [a,c,*], Jaesung Lee [b,c,*]

[a] School of Computer Science and Engineering, Chung-Ang University, 221, Heukseok-Dong, Dongjak-Gu, Seoul, 06974, Republic of Korea
[b] Department of Artificial Intelligence, Chung-Ang University, 221, Heukseok-Dong, Dongjak-Gu, Seoul, 06974, Republic of Korea
[c] AI/ML Innovation Research Center, Chung-Ang University, 221, Heukseok-Dong, Dongjak-Gu, Seoul, 06974, Republic of Korea

ARTICLE INFO

Keywords:
Human activity recognition
Convolutional neural network
Real-time systems
Time-series analysis

ABSTRACT

Human activity recognition is an essential task for human-centered intelligent systems such as healthcare and smart vehicles, which can be accomplished by analyzing time-series signals collected from sensors in wearable devices. In these applications, real-time response is vital because prompt action is necessary for urgent events such as an elderly person falling or driving while drowsy. Although recurrent neural networks have been widely used owing to their temporal modeling capabilities, recent studies have focused on convolutional neural networks (CNNs) that are suitable for real-time responses because they incur lower computational costs. However, CNNs with a manual design may fail to achieve optimal accuracy due to varying computational budgets with applications or devices. In this paper, we propose a novel design framework that uses a mathematical approach to derive a CNN architecture suitable for a given computational budget. As a result, we introduce a grouped temporal shift network (GTSNet) with the network architecture to be flexibly modified by predefining the theoretical computation cost. We demonstrate the effectiveness of our framework in experiments, achieving the best performance for well-known public benchmark datasets under limited computational budgets. The source codes of the GTSNet are publicly available at https://github.com/jgpark92/GTSNet.

1. Introduction

Human activity recognition (HAR) aims to identify human activities using time-series signals acquired from sensors (Dang et al., 2020). In recent years, studies have focused on deep neural networks (DNNs) because of their excellent performance (Huang et al., 2019; Wang et al., 2019). In real-time HAR, DNNs are generally trained on a high-performance server and then deployed to resource-limited edge devices, especially wearable devices, where inference is conducted (Ignatov, 2018; Zebin et al., 2019; Cheng et al., 2022). To achieve a real-time response, reducing the computational cost of DNNs is essential owing to their complex architectures and resource-limited devices (Dang et al., 2020; Wang et al., 2019; Chen et al., 2021). A straightforward solution to this issue is to reduce the number of network connections, which reduces the number of computations (Chollet, 2017; Howard et al., 2017; Wu et al., 2018; Zhang et al., 2018).

Recent real-time HAR studies have focused on one-dimensional (1D) convolutional neural networks (CNNs), which are more efficient than other networks such as recurrent neural networks (RNNs) (Ignatov, 2018; Wan et al., 2020). To achieve real-time HAR, lightweight networks were manually designed and evaluated by experimentally measuring their inference times for a specific device (Ignatov, 2018; Zebin et al., 2019; Cheng et al., 2022; Xu et al., 2019a). However, because the inference time of each network depends on the hardware specifications of the device (Xu et al., 2019b; Yu et al., 2018), hand-crafted networks should be redesigned manually when the target device changes. This redesign process requires considerable expert knowledge and time from practitioners because of the complex tradeoff between accuracy and computational cost.

Against this limitation, this study aims to establish a novel mathematical framework to re-design a network architecture according to a given computation budget. Specifically, our research questions include: *how can be the theoretical computation cost of the network architecture formalized?* and *how can be the network architecture flexibly modified according to the computation budget?*.

By answering the above questions, we propose a novel deep learning design framework for deriving a network architecture tailored to the given computational budget from a backbone network. The proposed framework can be used to easily modify a network architecture flexibly and directly using a mathematical approach instead of an exhaustive or labor-intensive approach. We first analyzed and formalized the

* Corresponding authors.
E-mail addresses: dwkim@cau.ac.kr (D.-W. Kim), curseor@cau.ac.kr (J. Lee).

https://doi.org/10.1016/j.engappai.2023.106543
Received 28 May 2022; Received in revised form 22 May 2023; Accepted 24 May 2023
Available online xxxx

theoretical computational cost of a backbone CNN with a familiar architecture, identifying the major factors that increase the defined cost. Subsequently, we introduced a grouped temporal shift (GTS) module that allows the network architecture to be flexibly modified by predefining the theoretical computational cost. In experiments, the proposed framework outperformed other networks.

The main contributions of this paper are as follows:

- We propose a novel design framework that uses a mathematical approach for reconstructing the network architecture without requiring exhaustive or labor-intensive manual redesign.
- We introduce the GTSNet that allows its architecture to be flexibly modified by predefining the theoretical computation cost to match the desired budget constraints across various wearable devices.
- We demonstrate the superiority of GTSNet in achieving the best performance for four well-known public benchmark datasets under limited computational budgets.

The remainder of this paper is organized as follows. In Section 2, we describe related works on efficient and mathematical approaches in terms of the real-time HAR. Section 3 clarifies the problem and goal of this paper. Section 4 presents the details of our GTSNet and a novel design framework. Section 5 elaborates on the experimental setup and results. Section 6 provides more insights and implications focusing on real-world applications. Finally, Section 7 concludes the paper and presents future steps.

2. Related work

Human activity can be regarded as a sequence of several continuous basic movements in which the transitions generate substantial changes within each signal datum (Chen et al., 2021). To capture the transitions that are closely related to the activities from the input signals, DNNs must be able to extract useful temporal features. RNNs, including recurrent units such as long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and bidirectional LSTM (Graves and Schmidhuber, 2005), have been used for modeling sequential data. In HAR, diverse RNN variants (Murad and Pyun, 2017; Guan and Plötz, 2017; Li et al., 2019) have achieved high accuracies because of their temporal modeling capabilities. Furthermore, Ordóñez and Roggen (2016) proposed DeepConvLSTM, combining CNNs and LSTMs to abstract and enhance features used for temporal modeling. However, RNNs incur infeasible computational costs for edge implementation (Zebin et al., 2019; Gehring et al., 2017), making real-time HAR difficult. Meanwhile, Fawaz et al. (2019) showed the potential of CNNs compared with other networks, including recurrent units or attention layers in the time series classification. Specifically, the temporal residual neural network (Wang et al., 2017) and the temporal fully-convolutional network (Wang et al., 2017) exhibited superior accuracy.

To the best of our knowledge, most studies have focused on improving the HAR accuracy (Xu et al., 2019a; Ordóñez and Roggen, 2016; Peng et al., 2018; Lv et al., 2019; Koli and Bagban, 2020), even though many comprehensive surveys in the HAR literature have emphasized the importance of real-time applications (Dang et al., 2020; Wang et al., 2019; Chen et al., 2021; Shoaib et al., 2015; Qi et al., 2018). Ravi et al. (2016) attempted real-time HAR by integrating short-time Fourier transform into a preprocessing step. They transformed the signal data from the time domain to the frequency domain and then fed the obtained spectral image into a two-dimensional (2D) CNN; the frequency axis was processed as channels. Bhat et al. (2018) proposed a real-time HAR framework including preprocessing, e.g., a discrete wavelet transform and DNN based on a policy gradient update. However, these approaches require complex preprocessing that should be conducted without stopping, increasing the overhead.

To resolve these issues, recent studies on real-time HAR have focused on 1D CNNs, as they incur lower computational costs than RNNs. Ignatov (2018) proposed a CNN architecture consisting of one convolutional layer and one fully-connected layer. It manually extracted simple statistical features and fed them into a fully-connected layer to increase the accuracy. Wan et al. (2020) proposed a CNN architecture comprising three convolutional layers and two fully-connected layers. According to their experimental results, the proposed CNN outperformed RNN variants with regard to the accuracy, despite its high efficiency. Also, Oluwalade et al. (2021) compared the performance of CNN to three RNN variants on smartphones and smartwatches sensor data. Similar to Wan et al. (2020), the networks including the convolutional layers outperformed other networks. In addition, Zebin et al. (2019) demonstrated the efficiency of parameter quantization as postprocessing for further optimization of the CNN.

Besides, designing efficient lightweight CNNs has been extensively studied in computer vision. Squeezenet (Iandola et al., 2016) significantly reduced the network connections by squeezing the number of channels based on pointwise convolutions. ResNext (Xie et al., 2017) showed superior performance over the number of parameters by adopting a split-transform-merge strategy. MobileNet (Howard et al., 2017) introduced a depthwise separable convolution that deconstructs the standard convolution to depthwise and pointwise convolution, resulting in $\sim 9\times$ fewer computations. To reduce computations of the pointwise convolution, ShuffleNet (Zhang et al., 2018) introduced a group convolution. Furthermore, MobileNetV2 (Sandler et al., 2018) reduced the memory footprint based on inverted residuals. In addition, Wu et al. (2018) replaced depthwise convolution with shift operation without deriving any FLOPs and parameters while maintaining accuracy.

In summary, the existing lightweight CNNs were manually designed to achieve a real-time response for a specific device. However, in practice, these manual architectures often need to be redesigned to optimize the accuracy under the computational budgets for various wearable devices, because the allowed computational costs of the devices for ensuring a real-time response can differ. One promising strategy for resolving this issue is as follows: (1) devising a flexible network architecture that can be modified according to a specific computational budget, (2) measuring the computational budget by counting basic operations processed on the target device during the response time required by the target application (Zhang et al., 2018; Ding et al., 2021), and (3) adjusting the network architecture such that it is suitable for the given computational budget.

Meanwhile, there are various mathematical approaches with neural networks in widespread applications, such as efficient energy management systems (Ghadami et al., 2021; Shahsavar et al., 2021) and electrical discharge machines (Gholizadeh et al., 2022). These studies are orthogonal with our study because they focus more on enhancing their system than network architecture. Our study focuses on deriving an efficient neural architecture based on a mathematical approach for real-time activity recognition.

3. Problem statement

In this section, we describe the mathematical redesign process of the CNN architecture. Let $C_n(\cdot)$ be the computational cost of CNNs. Given a well-established backbone network $\mathcal{A}$ and a computation budget C_b, our goal is to derive a network $\mathcal{A}^*$ satisfying the following condition:

$$C_n(\mathcal{A}^*) \lessapprox C_b < C_n(\mathcal{A}) \tag{1}$$

Fig. 1 shows a concept of our framework to be achieved in this paper. To this end, we need to define $C_n(\cdot)$ and devise a procedure to derive the architecture $\mathcal{A}^*$ from $\mathcal{A}$, which corresponds to (a) and (b) in Fig. 1, respectively. Therefore, in the next section, we formalize the network's theoretical computation cost and introduce a novel module that allows the network architecture to be flexibly modified by predefining the cost.

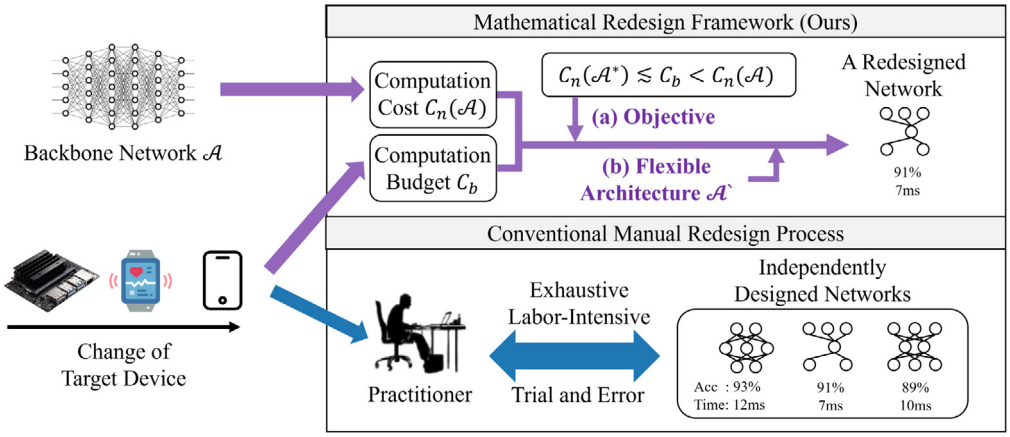


Fig. 1. Goal of a new deep-learning design framework. Conventional manual network design methods require considerable expert knowledge and time from practitioners. Contrarily, our framework uses a mathematical approach that can quickly adapt to the computation budget change by deriving a network architecture from the backbone network based on a theoretical computation cost.

Table 1
Notations used for describing the proposed framework.

Symbol	Meanings
T	Temporal resolution of an input for each convolution
M	Number of input channels for each convolution
N	Number of output channels for each convolution
I	Input for each convolution, $I \in \mathbb{R}^{T,M}$
O	Output for each convolution, $O \in \mathbb{R}^{T,N}$
$\mathbb{T}$	Temporal resolution of an input for a network
$\mathbb{M}$	Number of input channels for a network
$\mathbb{I}$	Input for a network, $\mathbb{I} \in \mathbb{R}^{\mathbb{T},\mathbb{M}}$
$\hat{\mathbb{M}}$	Number of output channels for the first convolution
$\mathbb{N}$	Number of output channels for the first residual block
D_k	Kernel size of each convolution
$\mathbb{D}_k$	Fixed kernel size across convolutional layers
L	Number of layers or operations
K	Convolution kernel
V	Number of activities
G	Number of channel groups

4. Proposed method

In this section, we describe the developed novel deep-learning design framework that can be used to derive a network architecture tailored to the given computational budget from a backbone network. The allowed computational cost for ensuring a real-time response can differ among different target devices. Therefore, we developed a strategy for designing an elaborate network architecture under a given computational budget to optimize the accuracy while ensuring a real-time response. Table 1 presents the terms used in this study.

This section is organized as follows. Section 4.1 describes how to formalize the theoretical computation cost of the backbone network. After that, Section 4.2 introduces our grouped temporal shift (GTS) module. Lastly, Section 4.3 describes how to modify the network architecture according to the computation budget by using the GTS module.

4.1. Major factor of backbone network

To analyze and formalize the computational cost at the network level, we introduced a backbone network based on an existing network with a familiar architecture. One candidate backbone network is the temporal residual neural network (T-ResNet) (Wang et al., 2017; Fawaz et al., 2019), which has achieved comparable performance in many time-series classification applications (Karim et al., 2017; Wang et al., 2018; Franceschi et al., 2019; Dempster et al., 2020; Shifaz et al., 2020). T-ResNet has three residual blocks, each of which is composed of three convolutional layers and a residual connection. It uses global average pooling (GAP) and a fully-connected layer for classification. The computational cost of T-ResNet is formulated in Proposition 1.

Proposition 1. *The computational cost of T-ResNet is formalized as*

$$C_n = \underbrace{\sum_{l=1}^{L-1} \alpha(\mathcal{K}^{(l)}(I^{(l)};\theta^{(l)}))}_{\text{Part 1}} + \underbrace{\alpha(\mathcal{F}(I^{(L)};\theta^{(L)}))}_{\text{Part 2}}. \tag{2}$$

Proof. T-ResNet consists of $L-1$ convolutional layers $\mathcal{K}^{(l)}$ and a fully-connected layer $\mathcal{F}$. For each layer, the input $I^{(l)}$ is fed into the lth layer with kernel parameters $\theta^{(l)}$ to calculate the output $I^{(l+1)}$. Let $\alpha(\cdot)$ be the computational cost of calculating the output of each layer. Because the computational cost of neural networks is determined by the layer composition, the computational cost of T-ResNet is formulated as $\sum_{l=1}^{L-1} \alpha(\mathcal{K}^{(l)}(I^{(l)};\theta^{(l)})) + \alpha(\mathcal{F}(I^{(L)};\theta^{(L)}))$. □

To formalize the computational cost of the network, we borrowed the concept of time complexity as the upper bound of the cost. As T-ResNet uses a standard convolution (StandardConv) (Fukushima and Miyake, 1982), part 1 of Eq. (2) can be replaced with the time complexity of StandardConv. Thus, Proposition 1 is simplified to Proposition 2.

Proposition 2. *The time complexity of T-ResNet is formalized as*

$$\underbrace{\sum_{l=1}^{L-1} \mathcal{O}(T^{(l)} D_k^{(l)} M^{(l)} N^{(l)})}_{\text{Part 3}} + \mathcal{O}(N^{(L-1)} V). \tag{3}$$

Proof. Let $I \in \mathbb{R}^{T,M}$ and $O \in \mathbb{R}^{T,N}$ be the input and output for StandardConv, respectively, where T represents the temporal resolution, and M and N represent the numbers of channels. Given an input and a kernel $K \in \mathbb{R}^{D_k,M,N}$ with kernel size D_k, the nth channel of O at time t is calculated as follows:

$$O_{t,n} = \sum_{i,m} K_{i,m,n} I_{t+\hat{i},m}, \tag{4}$$

where $\hat{i} = i - \lfloor D_k/2 \rfloor$ is the re-centered time index. Thus, the time complexity of StandardConv is $\mathcal{O}(T D_k M N)$.

Because T, D_k, M, and N can vary among the layers, part 1 of Eq. (2) can be replaced with $\sum_{l=1}^{L-1} \mathcal{O}(T^{(l)} D_k^{(l)} M^{(l)} N^{(l)})$. In addition, T-ResNet includes GAP immediately before the fully-connected layer for classification. Given the number of activities V, part 2 of Eq. (2) can be replaced with $\mathcal{O}(N^{(L-1)} V)$. Consequently, the time complexity of T-ResNet is formalized by Eq. (3). □

To simplify part 3 of Eq. (3), we added one StandardConv at the beginning of the network to fix the number of input channels to the first residual block because it varies depending on the number of sensors or their type. Therefore, the upper bound of the summation in part 3 of Eq. (3) becomes L without a change in the total time complexity. In addition, we fixed the kernel size and inserted max-pooling layers as the number of channels increased to formulate the relationship between the input and output sizes. The linear variations in D_k and T have no effect on the time complexity of each convolution. Thus, Eq. (3) is simplified by Proposition 3.

Proposition 3. *The time complexity of the backbone network is formalized as*

$$\mathcal{O}(\mathbb{T}\mathbb{D}_k\mathbb{N}^2 L). \tag{5}$$

Proof. Given the original signals $\mathbb{I} \in \mathbb{R}^{\mathbb{T},\mathbb{M}}$, the first StandardConv generates new feature signals $I^{(1)} \in \mathbb{R}^{\mathbb{T},\hat{\mathbb{M}}}$, resulting in a time complexity of $\mathcal{O}(\mathbb{T}\mathbb{D}_k\mathbb{M}\hat{\mathbb{M}})$. Suppose that the lth intermediate convolutional layer includes a kernel $K^{(l)} \in \mathbb{R}^{\mathbb{D}_k,N^{(l-1)},N^{(l)}}$, where $\mathbb{D}_k$ is a fixed kernel size across convolutional layers, and $N^{(l)}$ is a positive-integer multiple of $N^{(l-1)}$, that is, $N^{(l)} = c^{(l)}N^{(l-1)}$. Generally, $c \geq 1$; hence, the pooling layer adjusts the temporal resolution T to $\frac{T}{c}$ if $c \geq 2$. Therefore, its time complexity is $\mathcal{O}\left(\frac{T}{c}\mathbb{D}_k N^{(l-1)} c N^{(l-1)}\right) = \mathcal{O}(T\mathbb{D}_k(N^{(l-1)})^2)$. As $\mathcal{O}(T^{(l-1)}D_k^{(l-1)}(N^{(l-1)})^2) = \mathcal{O}(T^{(l)}D_k^{(l)}(N^{(l)})^2)$, the total time complexity of the intermediate convolutional layers becomes $\mathcal{O}(\mathbb{T}\mathbb{D}_k\mathbb{N}^2 L)$, eliminating the superscript l in Part 3 of Eq. (3).

In Proposition 2, the time complexity of T-ResNet is formulated by Eq. (3). By constructing the backbone network from T-ResNet, the time complexity can be simplified as follows:

$$\mathcal{O}(\mathbb{T}\mathbb{D}_k\mathbb{M}\hat{\mathbb{M}}) + \mathcal{O}(\mathbb{T}\mathbb{D}_k\mathbb{N}^2 L) + \mathcal{O}(N^{(L-1)}V). \tag{6}$$

Because $\mathbb{M}$ and V can be regarded as constant values in terms of the network architecture, Eq. (6) can be rewritten as $\mathcal{O}(\mathbb{T}\mathbb{D}_k(\hat{\mathbb{M}} + \mathbb{N}^2 L)) + \mathcal{O}(N^{(L-1)})$. Because $\hat{\mathbb{M}} \leq \mathbb{N}$ and $N^{(L-1)} \leq \mathbb{T}\mathbb{N}$, the time complexity of the backbone network becomes $\mathcal{O}(\mathbb{T}\mathbb{D}_k\mathbb{N}^2 L)$. □

Proposition 3 indicates that the time complexity of a network is closely related to the computational cost of each convolution. Therefore, we considered $\mathbb{D}_k$ and $\mathbb{N}$ to be the major factors related to extracting temporal and interchannel information (Chen et al., 2021). In this regard, StandardConv can be divided into temporal and interchannel convolutions. Given the kernel $K^{(t)} \in \mathbb{R}^{D_k,M}$ of the temporal convolution, the mth channel at time t is calculated as

$$\hat{O}_{t,m} = \sum_i \underbrace{K_{i,m}^{(t)}}_{\text{Part 4}} I_{t+\hat{i},m}. \tag{7}$$

Next, for a given kernel $K^{(c)} \in \mathbb{R}^{M,N}$ of the interchannel convolution, the nth channel is calculated as

$$O_{t,n} = \sum_m \underbrace{K_{m,n}^{(c)}}_{\text{Part 5}} \hat{O}_{t,m}, \tag{8}$$

where the time complexity is $\mathcal{O}(TM(D_k + N))$.

Inspired by the spatial shift operation (Wu et al., 2018), we focused on the connections of temporal convolution. The spatial shift operation shifts some channels in the spatial direction, and then the information between features at neighboring times is exchanged by the following 1×1 convolution. This idea can be easily applied to HAR by shifting channels in the temporal direction. Consequently, the time complexity of the convolution can be reduced by introducing a temporal shift convolution, in accordance with Lemma 1.

Lemma 1. *The time complexity of the temporal shift convolution is formalized as*

$$\mathcal{O}(TMN). \tag{9}$$

Proof. Given the kernel size D_k, the kernel of the temporal shift convolution $\tilde{K} \in \mathbb{R}^{D_k,M}$ can be written as

$$\tilde{K}_{i,m} = \begin{cases} \theta_m, & \textbf{if } i = i_m \\ 0, & \textbf{otherwise}, \end{cases} \tag{10}$$

where θ_m is a trainable parameter, and i_m is a channel-dependent index that only has one value in $\tilde{K}_{:,m} \in \mathbb{R}^{D_k}$.

For extracting temporal information, the temporal shift convolutions should be able to interact with each other between features of adjacent times. This can be achieved simply by spreading the i_m of Eq. (10) across adjacent times, while satisfying the following conditions:

$$\forall i : \sum_m^M |\tilde{K}_{i,m}| \neq 0. \tag{11}$$

Therefore, M channels are evenly divided into D_k parts, and i_m for each of them is assigned a different value. Thus, part 4 of Eq. (7) can be replaced with $\tilde{K}$. Additionally, the parameters of $\tilde{K}$ can be replaced with 1 by the associative law of multiplication between them and the parameters of part 5 of Eq. (8). Therefore, the temporal shift convolution has the same time complexity as the interchannel convolution, that is, $\mathcal{O}(TMN)$. □

4.2. Grouped temporal shift module

In this section, we introduce a novel GTS module to control network connections based on major factors. Although the temporal shift convolution allows the increase in $\mathbb{D}_k$ to have no effect on the network's time complexity, it suffers from overwhelming interchannel connections related to another major factor $\mathbb{N}$. To solve this issue, we first designed a new GTS convolution (GTSConv), and then the GTS module was constructed by addressing several issues of GTSConv.

A straightforward solution for reducing the number of interchannel connections is to divide the channels into multiple channel groups (Zhang et al., 2018). Fig. 2 shows a schematic of GTSConv. The M input channels are evenly divided into G channel groups. For each channel group, the channels are shifted in the temporal direction using Eq. (10). Subsequently, group-wise interchannel convolution generates N output channels in accordance with Eq. (8). Thus, the interchannel connections are controlled by the number of groups G, which is an additional factor used in Section 4.3, to derive a network architecture tailored to the computational budget based on time complexity. The time complexity of GTSConv is formulated by Lemma 2.

Lemma 2. *The time complexity of GTSConv is formalized as*

$$\mathcal{O}\left(\frac{TMN}{G}\right). \tag{12}$$

Proof. Given an input $I \in \mathbb{R}^{T,M}$ and the number of channel groups G, M channels are evenly divided into G channel groups, making the gth input channel group $I^{(g)} \in \mathbb{R}^{T,\lfloor M/G \rfloor}$. Subsequently, GTSConv shifts the input channels in the gth group by using $\tilde{K}$ of Eq. (10) as follows:

$$\tilde{I}_{t,m}^{(g)} = \sum_t \tilde{K}_{t,m,g} I_{t+\hat{i},m}^{(g)}, \tag{13}$$

where the computational cost is zero because the calculations are conducted by the following interchannel convolution, as described in Lemma 1. Finally, GTSConv generates the nth output channel in the gth group, as follows:

$$O_{t,n}^{(g)} = \sum_m K_{m,n,g}^{(c)} \tilde{I}_{t,m}^{(g)}, \tag{14}$$

where m and n range from 1 to $\lfloor M/G \rfloor$ and $\lfloor N/G \rfloor$, respectively. Because Eq. (14) is calculated for G channel groups, the time complexity of GTSConv is $\mathcal{O}\left(T \times \frac{M}{G} \times \frac{N}{G} \times G\right) = \mathcal{O}\left(\frac{TMN}{G}\right)$. □

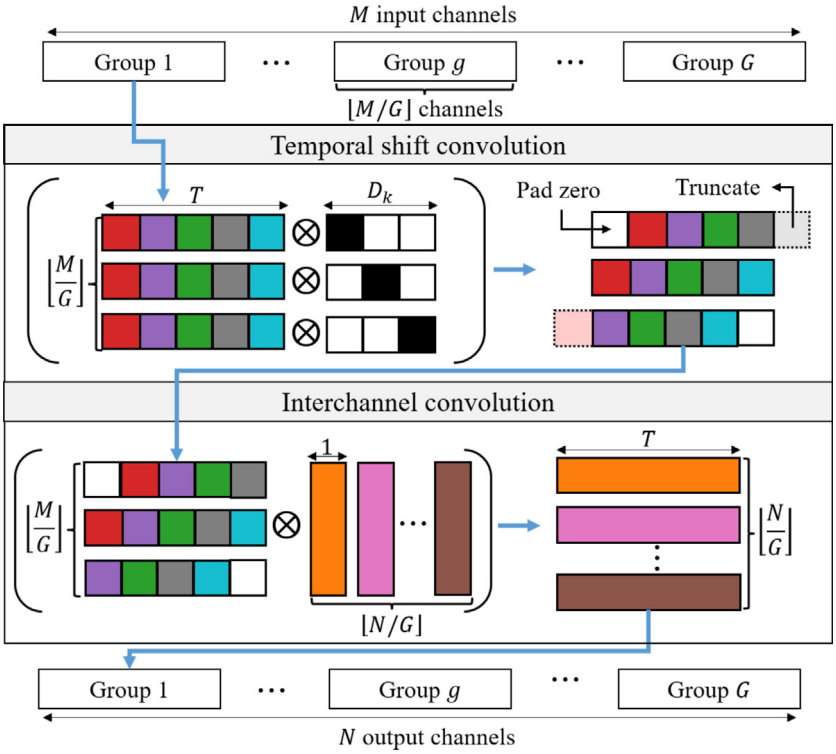


Fig. 2. Illustration of our GTS convolution. M input channels are evenly divided into G channel groups, and then each of the groups is fed into the temporal shift convolution and interchannel convolution, resulting in N output channels.

Furthermore, we addressed the following issues of GTSConv, resulting in the GTS module: first, the order of the channels within each channel group should be considered because shifting each channel in a better temporal direction can reduce the loss of information (Jeon and Kim, 2018). Second, it is unknown how to map M input channels to G channel groups in advance, because their correlations can vary whenever the trainable parameters are updated with stochastic gradient descent. To address these issues, permutation of the input channels can be considered.

Let $\mathcal{P}$ be a permutation function for the arrangement of the channels within each group. Accordingly, GTSConv can be rewritten as

$$\begin{aligned} O^{(g)} &= (\mathcal{K} \circ \mathcal{P})(I^{(g)}) \\ &= \mathcal{K}(I^{(g)} P), \end{aligned} \tag{15}$$

where $\circ$ denotes function composition, and $\mathcal{K}$ is computed using Eqs. (13) and (14). The permutation function $\mathcal{P}$ can be conducted by multiplying a permutation matrix $P \in \mathbb{R}^{\lfloor M/G \rfloor, \lfloor M/G \rfloor}$ to $I^{(g)} \in \mathbb{R}^{T, \lfloor M/G \rfloor}$. However, because P has discrete values, it is difficult to optimize P using stochastic gradient descent. A straightforward way to resolve this issue is to train the interchannel convolution directly for each channel group (Wu et al., 2018). Specifically, each input channel group is permuted by conducting an additional interchannel convolution before GTSConv is applied.

However, since P has discrete values, it is difficult to optimize P by stochastic gradient descent. A straightforward way to resolve this issue can be to train the inter-channel convolution directly for each channel group (Wu et al., 2018). Specifically, each input channel group is permuted by conducting an additional inter-channel convolution before GTSConv is conducted.

Although a similar mapping of channel groups can be solved, the interchannel convolution for the overall channels has a higher time complexity than GTSConv. Therefore, we used the channel shuffle operation of ShuffleNet (Zhang et al., 2018). Specifically, if the channels extracted from each GTSConv layer are randomly shuffled and delivered to the next GTSConv layer, the entire channel will be fully related as the layers are stacked.

Algorithm 1 Grouped Temporal Shift module

1: **Input:** $I \in \mathbb{R}^{T,M}$, G; ▷ number of channel groups G
2: **Output:** $O \in \mathbb{R}^{T,N}$;
3: **for** $g = 1$ to G **do**
4: $[I^{(1)}, ..., I^{(g)}, ..., I^{(G)}] \leftarrow$ divide channels of I into G groups evenly;
5: $\hat{I}^{(g)} \leftarrow$ calculate **Permutation**$(I^{(g)}, \lfloor N/G \rfloor)$ ▷ use Eq. (8)
6: $\hat{I}^{(g)} \leftarrow \text{ReLU}(\text{BN}(\hat{I}^{(g)}))$ ▷ batch normalization BN
7: $\tilde{O}^{(g)} \leftarrow$ calculate **GTSConv**$(\hat{I}^{(g)}, D_k)$ ▷ use Eqs. (13) and (14)
8: $\tilde{O}^{(g)} \leftarrow \text{ReLU}(\text{BN}(\tilde{O}^{(g)}))$
9: **end for**
10: $O \leftarrow$ **Concatenation**$(\tilde{O})$ ▷ concatenate them at channel axis
11: $O \leftarrow$ **Shuffle**(O) ▷ shuffle them at channel axis

Finally, Algorithm 1 presents the pseudocode for the forward pass of the GTS module. In Lines 5–8, each input channel group is permuted by the interchannel convolution and then fed into GTSConv, and all convolutions come with batch normalization and rectified linear unit. The shuffle operation can be used at the starting point or endpoint of each module by stacking the GTS modules. Generally, $M \leq N$; thus, the shuffle operation is conducted at the endpoint to enhance its effectiveness (Line 11). The time complexity of the GTS module is determined by permutation and GTSConv in Lines 5 and 7. As they have identical time complexity to $\mathcal{O}\left(\frac{TMN}{G}\right)$, our GTS module has a time complexity of $\mathcal{O}\left(2 \times \frac{TMN}{G}\right) = \mathcal{O}\left(\frac{TMN}{G}\right)$. Finally, we introduced GTSNet by replacing all the intermediate StandardConv layers of the backbone network with the GTS modules, as shown in Fig. 3. Specifically, the network architecture is a simple variant of T-ResNet, designed by adding a first StandardConv and max-pooling layers as described

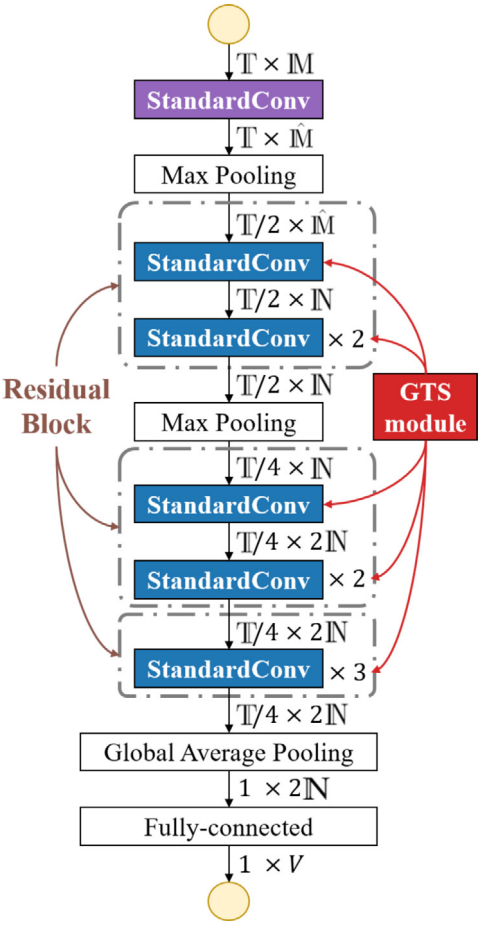


Fig. 3. Neural architecture of the backbone network and GTSNet. GTSNet is constructed by replacing StandardConv (blue boxes) with the proposed GTS module.

in Section 4.1. Thus, Eq. (5) in Proposition 3 is reduced to that in Lemma 3.

Lemma 3. *The time complexity of GTSNet is formalized as*

$$\mathcal{O}\left(\frac{\mathbb{T}\mathbb{N}^2 L}{G}\right). \tag{16}$$

Proof. Similar to Proposition 3, the time complexity of GTSNet is

$$\mathcal{O}(\mathbb{T}\mathbb{D}_k\mathbb{M}\hat{\mathbb{M}}) + \mathcal{O}\left(\frac{\mathbb{T}\mathbb{N}^2}{G}L\right) + \mathcal{O}(N^{(L-1)}V), \tag{17}$$

where the first term represents the time complexity of the first StandardConv, the second term represents the time complexity of the GTS modules, and the third term represents the time complexity of the fully-connected layer.

Because $\mathbb{M}$ and V can be regarded as constant values in terms of the network architecture, Eq. (17) can be rewritten as follows:

$$\mathcal{O}(\mathbb{T}\mathbb{D}_k\hat{\mathbb{M}}) + \mathcal{O}\left(\frac{\mathbb{T}\mathbb{N}^2}{G}L\right) + \mathcal{O}(N^{(L-1)}). \tag{18}$$

Given $\hat{\mathbb{M}} \leq \mathbb{N}$ and $N^{(L-1)} \leq \mathbb{T}\mathbb{N}$, Eq. (18) is rewritten as

$$\mathcal{O}\left(\mathbb{T}\mathbb{N}\left(\mathbb{D}_k + \frac{\mathbb{N}}{G}L\right)\right), \tag{19}$$

where $\frac{\mathbb{N}}{G}$ represents the number of channels in each channel group. The condition of Eq. (11) can be satisfied if and only if $\frac{\mathbb{N}}{G} \geq \mathbb{D}_k$. As $\mathbb{D}_k > 2$ and $L \geq 2$, $\frac{\mathbb{N}}{G}L \gg \mathbb{D}_k$. Therefore, the time complexity for the backbone network is reduced to Eq. (16) by replacing all intermediate StandardConv with the GTS modules. □

4.3. Complexity-based network design strategy

In this section, we introduce a novel strategy for controlling the network connections according to the time complexity to achieve the optimal accuracy under the computational budget. We consider $\mathbb{N}$ to be a major factor because GTSNet suffers from a factor that grows quadratically in its time complexity, that is, $\mathcal{O}(\mathbb{N}^2)$. If a theoretical computational cost is predefined according to a given computational budget, our framework can rapidly achieve a final network by narrowing the range of possible architectures. Let $\mathcal{O}(\mathbb{T}LU)$ be the predefined cost, i.e., the target time complexity (TTC), where $\mathcal{O}(U) \leq \mathcal{O}(\mathbb{N}^2)$. We formalize the TTC using GTSNet's additional factor G, in accordance with Theorem 1.

Theorem 1. *The time complexity of GTSNet can be reduced to TTC of $\mathcal{O}(\mathbb{T}LU)$ if G, satisfying the following condition, exists:*

$$\exists G \in \mathbb{Z}^+ : \frac{\mathbb{N}^2}{U} \leq G \leq \frac{\mathbb{N}}{\mathbb{D}_k}, \tag{20}$$

where $\mathcal{O}(U) \leq \mathcal{O}(\mathbb{N}^2)$.

Proof. The quadratic growth of the time complexity comes from the interchannel convolution. In our GTS module, $\mathcal{O}(\mathbb{N}^2)$ becomes $\mathcal{O}\left(\frac{\mathbb{N}}{G} \times \frac{\mathbb{N}}{G} \times G\right)$ when the channels are divided into groups. Because $\frac{\mathbb{N}}{G} \geq \mathbb{D}_k$, the positive integer G should be less than or equal to $\frac{\mathbb{N}}{\mathbb{D}_k}$. Suppose that there is a value of G satisfying $\frac{\mathbb{N}^2}{G} \leq U$, where $\mathcal{O}(U) \leq \mathcal{O}(\mathbb{N}^2)$. In Lemma 3, the time complexity in Eq. (12) can be reduced to $\mathcal{O}(\mathbb{T}LU)$ by setting G according to Eq. (20). □

From Theorem 1, the conditions that satisfy Eq. (20) depend on the TTC of $\mathcal{O}(\mathbb{T}LU)$, where $\mathcal{O}(U) \leq \mathcal{O}(\mathbb{N}^2)$. There are several possible options for the TTC. The basic option is $\mathcal{O}(\mathbb{T}L\mathbb{N}^2)$, in accordance with Corollary 1.1.

Corollary 1.1. *GTSNet with $\mathcal{O}(\mathbb{T}L\mathbb{N}^2)$ can be achieved by satisfying the following condition:*

$$\mathbb{D}_k \leq \mathbb{N}. \tag{21}$$

Proof. When $U = \mathbb{N}^2$, the condition of Eq. (20) can be rewritten as

$$1 \leq G \leq \frac{\mathbb{N}}{\mathbb{D}_k}. \tag{22}$$

Therefore, a value of G satisfying the condition in Eq. (22) always exists if $\mathbb{D}_k \leq \mathbb{N}$. □

The number of input and output channels is larger than the kernel size in almost all the convolutional layers of the existing deep learning models, satisfying Eq. (21). Therefore, $\mathcal{O}(\mathbb{T}L\mathbb{N}^2)$ can be regarded as the basic option for the TTC. Next, consider the TTC to be $\mathcal{O}(\mathbb{T}L\mathbb{N}\log(\mathbb{N}))$. To achieve this, the relationship between $\mathbb{D}_k$ and $\mathbb{N}$ can be formulated as in Corollary 1.2.

Corollary 1.2. *GTSNet with $\mathcal{O}(\mathbb{T}L\mathbb{N}\log(\mathbb{N}))$ can be achieved by satisfying the following condition:*

$$\mathbb{D}_k < \frac{a \times 2^a}{a + 2^a}, \quad \mathbb{N} = 2^a \geq 2^3. \tag{23}$$

Proof. When $U = \mathbb{N}\log(\mathbb{N})$, the condition in Eq. (20) can be satisfied if $\frac{\mathbb{N}}{\mathbb{D}_k} - \frac{\mathbb{N}}{\log(\mathbb{N})} > 1$. Because $\mathbb{N}$ is commonly set to two powers, that is, $\mathbb{N} = 2^a$, $\mathbb{D}_k$ and $\mathbb{N}$ should be set to values that satisfy the following condition:

$$\begin{aligned} 1 &< \frac{2^a}{\mathbb{D}_k} - \frac{2^a}{a} \\ \mathbb{D}_k &< \frac{a \times 2^a}{a + 2^a} = U(a). \end{aligned} \tag{24}$$

For extracting temporal information, $\mathbb{D}_k$ should be ≥ 2. When $\mathbb{D}_k = 2$, a should be ≥ 3, according to Eq. (24). As $U(a)$ is an increasing function, Eq. (20) is satisfied if $a \geq 3$. □

Finally, we consider the TTC to be $\mathcal{O}(\mathbb{T}L\mathbb{N})$. To achieve this, a value of G satisfying $\frac{\mathbb{N}^2}{G} \leq \mathbb{N}$ should exist, but this violates the condition of Eq. (20). Instead, we consider the lowest time complexity that can be achieved in GTSNet. Because a higher G incurs a lower computational cost, the optimal efficiency of GTSNet is achieved, as indicated by Corollary 1.3.

Finally, we consider the TTC to be $\mathcal{O}(\mathbb{T}L\mathbb{N})$. To achieve this, G satisfying $\frac{\mathbb{N}^2}{G} \leq \mathbb{N}$ should exist, but it violates the condition of Eq. (20). Instead, we consider the lowest time complexity that can be achieved in GTSNet. Since the higher G incurs a lower computational cost, the best efficiency of GTSNet is achieved as in Corollary 1.3.

Corollary 1.3. *GTSNet achieves the optimal efficiency with $\mathcal{O}(\mathbb{T}L\mathbb{N})$ by satisfying the following condition:*

$$G = \frac{\mathbb{N}}{\mathbb{D}_k}. \tag{25}$$

Proof. As a higher G incurs a lower computational cost, we consider $\frac{\mathbb{N}}{\mathbb{D}_k}$ as the upper bound of G according to Eq. (20) of Theorem 1. From Lemma 3, the time complexity of GTSNet can be reduced to $\mathcal{O}(\mathbb{T}L\mathbb{D}_k\mathbb{N})$ by replacing G with $\frac{\mathbb{N}}{\mathbb{D}_k}$. To achieve the optimal efficiency for GTSNet, $\mathcal{O}(\mathbb{D}_k\mathbb{N}) < \mathcal{O}(\mathbb{N}\log(\mathbb{N}))$ should always be satisfied. From Corollary 1.2, Eq. (24) can be rewritten as

$$\mathbb{D}_k < \frac{\mathbb{N}\log(\mathbb{N})}{\mathbb{N} + \log(\mathbb{N})} < \log(\mathbb{N}). \tag{26}$$

Therefore, the optimal efficiency of GTSNet can be achieved with time complexity of $\mathcal{O}(\mathbb{T}L\mathbb{N})$. □

5. Experimental results

In this section, we verify the performance of our GTSNet. Section 5.1 describes the experimental settings, including benchmark datasets, evaluation metrics, baseline networks, and implementation details. Section 5.2 shows the compared results of GTSNet and other networks. Lastly, Section 5.3 investigates the effect of components of GTSNet on performance.

5.1. Experimental settings

Experiments were conducted using four public benchmark datasets. Details are presented below.

- The **UCI-HAR** dataset (Anguita et al., 2013) was aggregated from 30 subjects by using accelerometers and gyroscopes embedded in Android smartphones. Each subject performed six activities: "walking", "upstairs", "downstairs", "sitting", "standing", and "lying", The dataset was sampled at a frequency of 50 Hz, and the length of each segment was 128. As recommended by the authors, 70% and 30% of the dataset were used as the training and test sets, respectively.
- The **WISDM** dataset (Kwapisz et al., 2011) was aggregated from 36 subjects by using accelerometers embedded in Android smartphones. Each participant performed six activities: "jogging", "walking", "upstairs", "downstairs", "sitting", and "standing". In Ignatov (2018), when each segment covered 3 s, the CNN achieved the best tradeoff between accuracy and efficiency. Therefore, we set the segment length to 60, because the data were sampled at a frequency of 20 Hz. The data collected from subjects 1–26 were used as the training set, and the other data were used as the test set.
- The **OPPORTUNITY** dataset (Chavarriaga et al., 2013) contained measurements for complex activities in a sensor-rich environment with on-body, object, and ambient sensors. We only considered on-body sensors for real-time HAR, including accelerometers and inertial measurement units (IMUs); the number of input channels used in our experiments was 113. The dataset was sampled at a frequency of 30 Hz, aggregated from four subjects, and contained 18 classes, such as "open door" and "drink from cup". For each subject, the dataset contained ADL1–5 and Drill; the ADL3–4 data for subjects 3 and 4 were used as the test set, and the other data were used as the training set. The segment length was set to 150.
- The **PAMAP2** dataset (Reiss and Stricker, 2012) was aggregated from nine subjects who wore three IMUs at 100 Hz and a heart rate monitor at 9 Hz. The dataset consisted of 18 activities, including basic activities such as "walking" and complex activities such as "playing soccer". The authors recommended a sliding window of 5.12 s, and we set the segment length to 512. The datasets obtained by subjects 102 and 106 were used as the test set. Because "watching TV", "car driving", and "playing soccer" were performed by only one subject, 30% of the data were randomly added to the test set. The remaining data were used as the training set.

Three metrics were used to evaluate the performance of GTSNet: multiply-accumulate (MAC) operation, model size, and F1-score.

- The **MAC operation** is a basic computation that includes one multiplication operation and one addition operation. Therefore, we used MACs to measure the actual computational cost of neural networks.
- The **Model size** represents the number of parameters in a neural network. This metric is related to memory access, which affects the inference time.
- The **F1-score** has been commonly used as an alternative to accuracy in HAR studies. This is because the HAR datasets inherently involve a class imbalance. The F1-score is computed as follows:

$$F_1 = \sum_i 2w_i \frac{precision_i \times recall_i}{precision_i + recall_i}, \tag{27}$$

where w_i represents the proportion of patterns in ith class.

We compared GTSNet with eight baseline networks, as follows:

- **Real-time HAR models.** Ignatov (2018) introduced a CNN consisting of a single convolutional layer and a fully-connected layer. Additionally, Wan et al. (2020) proposed a CNN consisting of three convolutional layers and two fully-connected layers. These CNNs used a max-pooling layer that followed each convolution to reduce the temporal resolution. Their experimental results can be regarded as the baseline performance of real-time HAR models.
- **RNN Variants.** We adopted three RNNs widely used for modeling sequential data. The long short-term memory (LSTM) network consists of two recurrent layers with 128 LSTM units (Hochreiter and Schmidhuber, 1997) and a fully-connected layer. The bidirectional LSTM (BiLSTM) network consists of two BiLSTM (Graves and Schmidhuber, 2005) layers with 128 units and a fully-connected layer. The DeepConvLSTM network (Ordóñez and Roggen, 2016) includes four convolutional, two LSTM, and a fully-connected layer.
- **Time-Series Classification (TSC) models.** We adopted two CNNs that achieved significant success for the TSC problem, which is more general purpose than HAR. The temporal residual neural network (T-ResNet) (Wang et al., 2017; Fawaz et al., 2019) is described in Section 4.1, along with our backbone network. The temporal fully-convolutional network (T-FCN) (Wang et al., 2017; Fawaz et al., 2019) consists of three convolutional and a fully-connected layer. Compared with Wan et al. (2020), it has more output channels. These CNNs use a GAP layer, which averages channels across the time dimension, instead of other pooling layers.

Table 2
Comparison results for the UCI-HAR dataset. ▼/△ indicates that the corresponding model is significantly worse/better than the proposed network according to a paired t-test at a significance level of 95% for three metrics.

Model	MACs	Model size	F1 score
GTSNet (**Ours**)	2.996M	0.078M	0.957
Backbone network	11.220M▼	0.315M▼	0.954
Ignatov (2018)	8.950M▼	6.490M▼	0.925▼
Wan et al. (2020)	2.199M△	0.228M▼	0.939▼
LSTM	26.150M▼	0.203M▼	0.804▼
BiLSTM	69.076M▼	0.536M▼	0.914▼
DeepConvLSTM	33.602M▼	0.296M▼	0.891▼
T-ResNet	61.621M▼	0.482M▼	0.953
T-FCN	34.440M▼	0.269M▼	0.958
ResNext	27.466M▼	22.036M▼	0.941▼
MobileNetV2	11.997M▼	2.189M▼	0.925▼
ShuffleNet	5.634M▼	0.910M▼	0.937▼
SqueezeNet	5.338M▼	0.360M▼	0.920▼

Table 3
Comparison results for the WISDM dataset. ▼/△ indicates that the corresponding model is significantly worse/better than the proposed network according to a paired t-test at a significance level of 95% for three metrics.

Model	MACs	Model size	F1 score
GTSNet (**Ours**)	1.388M	0.078M	0.886
Backbone network	5.242M▼	0.315M ▼	0.884
Ignatov (2018)	3.658M▼	3.068M▼	0.840▼
Wan et al. (2020)	0.968M△	0.161M▼	0.862▼
LSTM	12.166M▼	0.201M▼	0.869▼
BiLSTM	32.196M▼	0.533M▼	0.862▼
DeepConvLSTM	13.312M▼	0.295M▼	0.877▼
T-ResNet	28.793M▼	0.480M▼	0.879▼
T-FCN	15.983M▼	0.267M▼	0.880▼
ResNext	12.878M▼	22.035M▼	0.869▼
MobileNetV2	5.972M▼	2.188M▼	0.861▼
ShuffleNet	2.811M▼	0.909M▼	0.870▼
SqueezeNet	2.146M▼	0.358M▼	0.845▼

- **Efficient CNNs** We used four CNNs (MobileNetV2 Sandler et al., 2018, ShuffleNet Zhang et al., 2018, SqueezeNet Iandola et al., 2016, and ResNext Xie et al., 2017) designed for computationally efficient image classification. We replaced 2D operations with 1D operations to conduct experiments on the HAR datasets.

All the networks were re-implemented in PyTorch (Paszke et al., 2019). For fairness, we excluded sophisticated tricks of each original setting, such as a gradient clipping (Mikolov et al., 2012) and learning rate warmup (He et al., 2019). We performed the experiments 10 times for all the datasets and obtained the average values; herein, a random cross-validation method was used on UCI-HAR and PAMAP2 datasets. We used Adam optimizer (Kingma and Ba, 2014) with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. Both the learning rate and weight decay were set to 0.0005. We trained the networks for 500 epochs with a batch size of 128 using a 2080Ti graphics processing unit. Each network was selected at the epoch that achieved the best performance during the training step and was applied to the test set.

For the backbone network, we set $\mathbb{D}_k$, $\mathbb{N}$, and $\hat{\mathbb{M}}$ to 3, 64, and 32, respectively. In GTSNet, $\mathbb{N}$ was set to 128, which was twice that of the backbone network, to mitigate accuracy degradation. In addition, we used the TTC of $\mathcal{O}(\mathbb{T}L\mathbb{N}\log(\mathbb{N}))$ and set G to 32, satisfying the condition of Eq. (20). The permutation in the first GTS module was conducted across $\hat{\mathbb{M}}$ overall input channels, because the condition of $\frac{\hat{\mathbb{M}}}{G} \geq \mathbb{D}_k$ may be unsatisfied when $\mathbb{N} \gg \hat{\mathbb{M}}$.

5.2. Comparison results

Tables 2–5 present the results of the comparison of GTSNet and the baseline networks. We performed the experiments 10 times and obtained the average values. For each dataset, we also conducted a paired t-test at the 95% significance level. In Tables 2–5, ▼/△ indicates that the corresponding network was significantly worse/better than GTSNet for the three metrics, respectively. As indicated by Tables 2 and 3, GTSNet (with the smallest model size) was statistically superior to the existing real-time HAR models, RNN Variants and efficient CNNs with regard to the F1-score for the UCI-HAR and WISDM datasets. More importantly, compared with T-ResNet, GTSNet significantly reduced the MACs and model size without degrading the F1-score.

As indicated by Tables 4 and 5, GTSNet (with the smallest number of MACs and smallest model size) outperformed the existing real-time HAR models with regard to the F1-score for the OPPORTUNITY and PAMAP2 datasets. Interestingly, despite its efficiency gain, GTSNet exhibited the same F1-score as the backbone network or a higher F1-score than the backbone network for all the datasets. Meanwhile, T-ResNet (with the largest number of MACs) achieved the best F1-score for the OPPORTUNITY dataset, as shown in Table 4. On the other hand,

Table 4
Comparison results for the OPPORTUNITY dataset. ▼/△ indicates that the corresponding model is significantly worse/better than the proposed network according to a paired t-test at a significance level of 95% for three metrics.

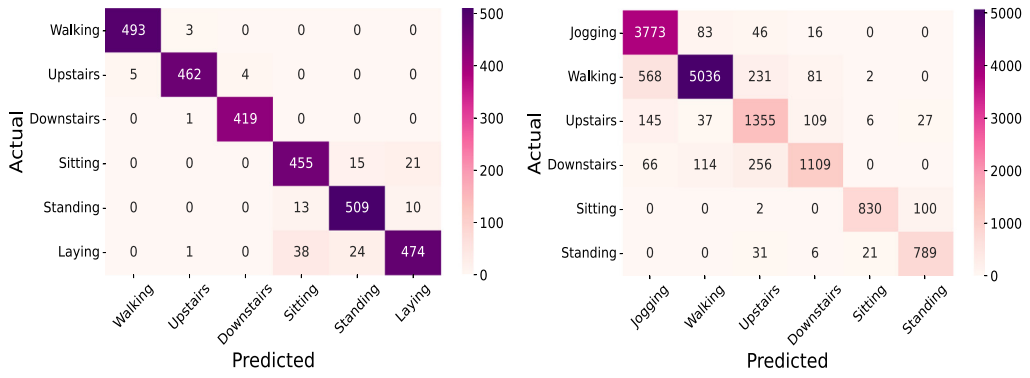
Model	MACs	Model size	F1 score
GTSNet (**Ours**)	5.022M	0.092M	0.874
Backbone network	14.550M▼	0.327M▼	0.876
Ignatov (2018)	31.507M▼	8.292M▼	0.871▼
Wan et al. (2020)	9.749M▼	0.315M▼	0.857▼
LSTM	38.863M▼	0.259M▼	0.845▼
BiLSTM	97.387M▼	0.649M▼	0.860▼
DeepConvLSTM	45.150M▼	0.332M▼	0.850▼
T-ResNet	80.431M▼	0.538M▼	0.883△
T-FCN	54.742M▼	0.367M▼	0.876
ResNext	32.461M▼	22.069M▼	0.877△
MobileNetV2	15.694M▼	2.214M▼	0.872
ShuffleNet	7.577M▼	0.929M▼	0.875
SqueezeNet	11.694M▼	0.438M▼	0.832▼

Table 5
Comparison results for the PAMAP2 dataset. ▼/△ indicates that the corresponding model is significantly worse/better than the proposed network according to a paired t-test at a significance level of 95% for three metrics.

Model	MACs	Model size	F1 score
GTSNet (**Ours**)	13.210M	0.084M	0.762
Backbone network	46.107M▼	0.319M▼	0.734▼
Ignatov (2018)	75.906M▼	25.930M▼	0.674▼
Wan et al. (2020)	14.375M▼	0.704M▼	0.718▼
LSTM	111.151M▼	0.217M▼	0.747▼
BiLSTM	289.412M▼	0.565M▼	0.684▼
DeepConvLSTM	151.946M▼	0.305M▼	0.773△
T-ResNet	253.037M▼	0.496M▼	0.714▼
T-FCN	149.228M▼	0.293M▼	0.725▼
ResNext	109.857M▼	22.053M▼	0.772△
MobileNetV2	48.596M▼	2.206M▼	0.759
ShuffleNet	22.991M▼	0.923M▼	0.798△
SqueezeNet	27.011M▼	0.383M▼	0.710▼

DeepConvLSTM outperformed our GTSNet for the PAMAP2 dataset with the largest segment length, as shown in Table 5. ShuffleNet, which had the largest number of channels, achieved the best F1-score for the PAMAP2 dataset, as shown in Table 5. These results indicate that it is essential to use sufficient channels to achieve a tradeoff between the computational cost and accuracy in real-time HAR.

To investigate the error of GTSNet, we also plot the confusion matrices in Figs. 4–6. Their diagonal elements represent the number of the patterns classified to correct activities, with the color lightening as the number grows. In Fig. 4, GTSNet with a fixed kernel size across layers tends to misclassify "walking" as "jogging" with a 10% error rate on



(a) UCI-HAR dataset (b) WISDM dataset

Fig. 4. Confusion matrix of GTSNet on UCI-HAR and WISDM dataset.

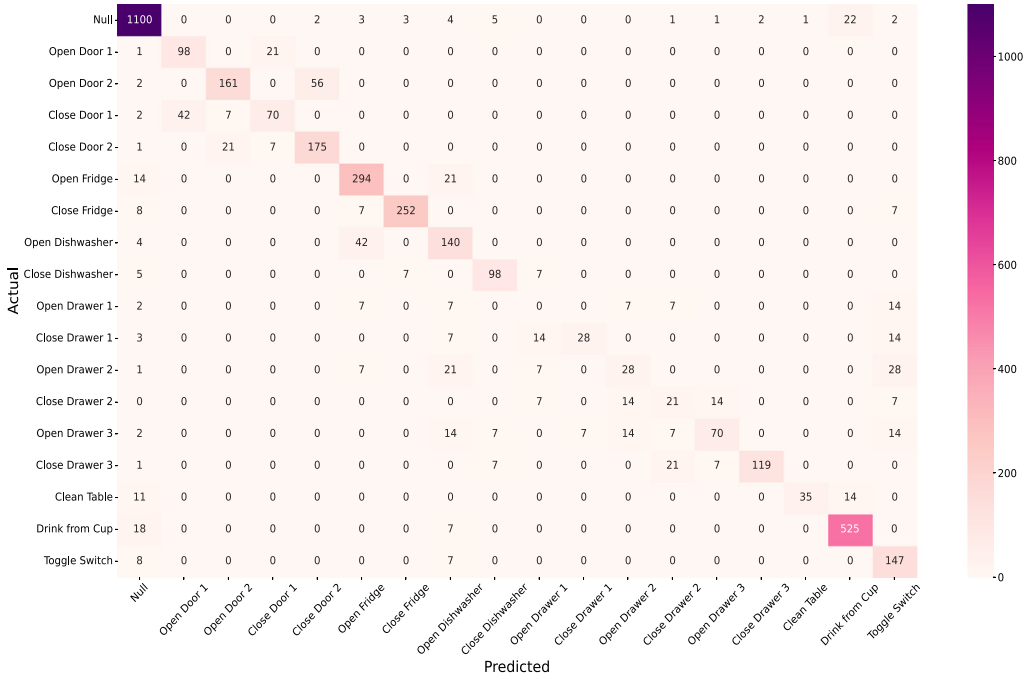


Fig. 5. Confusion matrix of GTSNet on OPPORTUNITY dataset.

the WISDM dataset. Also, GTSNet only with local temporal operations is efficient but tends to misclassify activities performed in the reverse order, such as "open door" and "close door" on the OPPORTUNITY dataset, as shown in Fig. 5. Furthermore, GTSNet may need more sophisticated architecture to correctly classify similar activities, such as "ironing" and "folding laundry" on the PAMAP2 dataset, as shown in Fig. 6. In conclusion, the confusion matrices indicate that our GTSNet has a low error rate for most activities across four HAR datasets.

5.3. Ablation study

Scalable architecture depending on $\mathbb{N}$. We compared the growth rate of the computational cost as the number of channels N increased among three networks: GTSNet, the backbone network, and the network of Wan et al. (2020). We set the TTC of GTSNet to $\mathcal{O}(\mathbb{T}L\mathbb{N})$. As shown in Fig. 7, GTSNet had an efficient rate of increase in MACs as $\mathbb{N}$ increased for all the datasets. This is because the time complexities of the backbone network and the network of Wan et al. grows by $\mathbb{N}$ quadratically. In addition, the MACs of the network of Wan et al. rapidly increased for the OPPORTUNITY dataset, with more input channels than the other datasets. This indicates the efficiency of scaling the number of input channels at the beginning of the network with fewer $\hat{\mathbb{M}}$ channels than $\mathbb{N}$.

F1-score vs. MACs. We investigated the tradeoff between the F1-score and MACs for the three networks. For the UCI-HAR and WISDM datasets (consisting of simple activities), we increased $\mathbb{N}$ according to Table 7 until the MACs reached a computational budget of 2^2M. For the OPPORTUNITY and PAMAP2 datasets (consisting of complex activities), we increased $\mathbb{N}$ until the MACs reached a computational budget of 2^4M. Fig. 8 shows the tradeoff between the MACs and F1-score for four HAR datasets; here, tradeoff curves closer to the top-left are more efficient, with a higher F1-score per MAC. As shown in Fig. 8, GTSNet achieved the highest F1-score under the various computational budgets of 2^a MACs for all the datasets. These results indicate that GTSNet achieved the highest accuracy under the given computational budget.

The breakdown effect of GTS module. Finally, we conducted the ablation study on our GTS module. Our GTS module is composed of the grouped shift convolution, shuffle across channel groups, and permutation within each channel group, as shown in Algorithm 1. Therefore, we compared the performance of our GTSNet with three variants of the GTSNet obtained by removing its components one by one. As

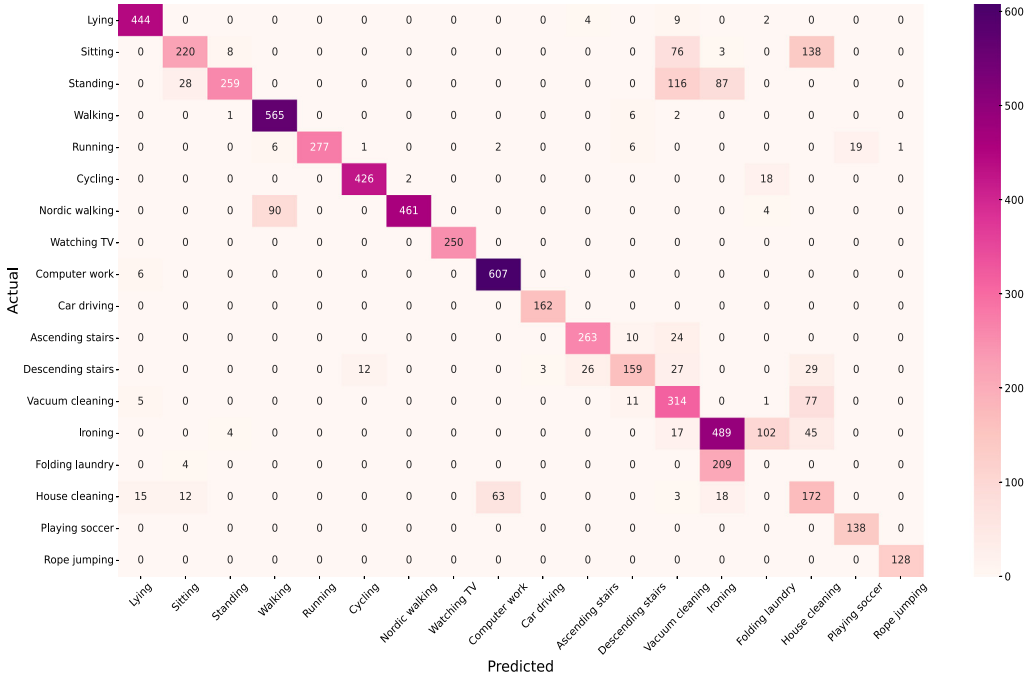
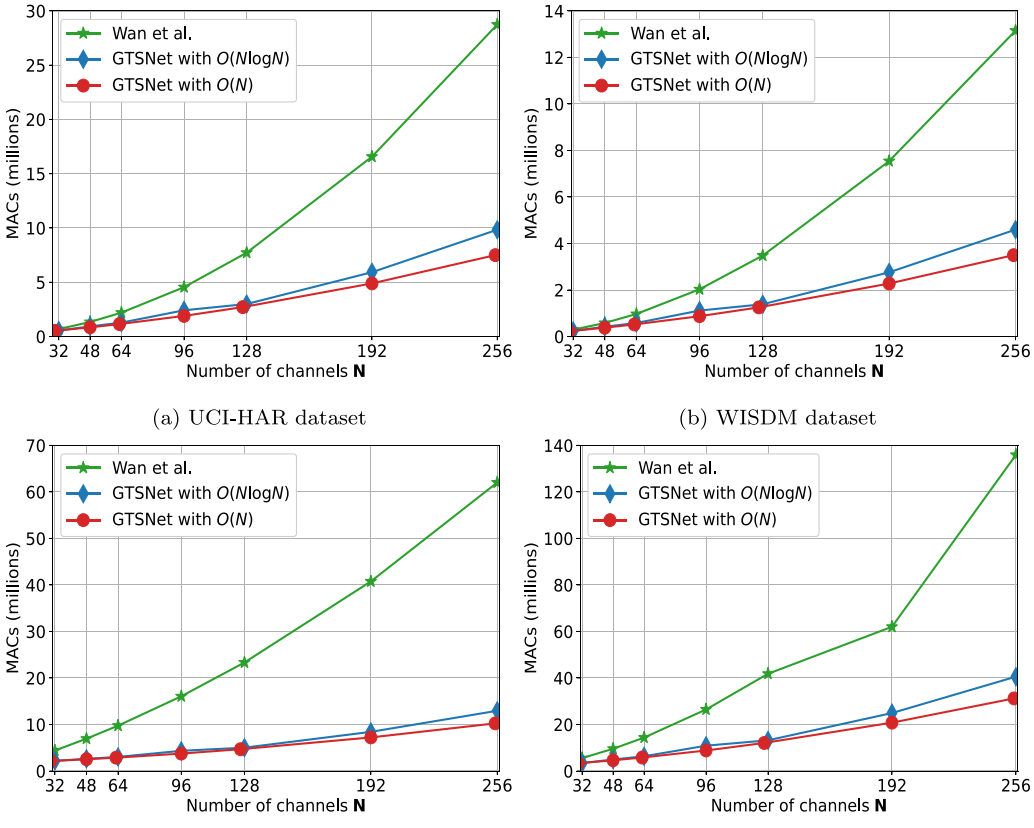


Fig. 6. Confusion matrix of GTSNet on PAMAP2 dataset.



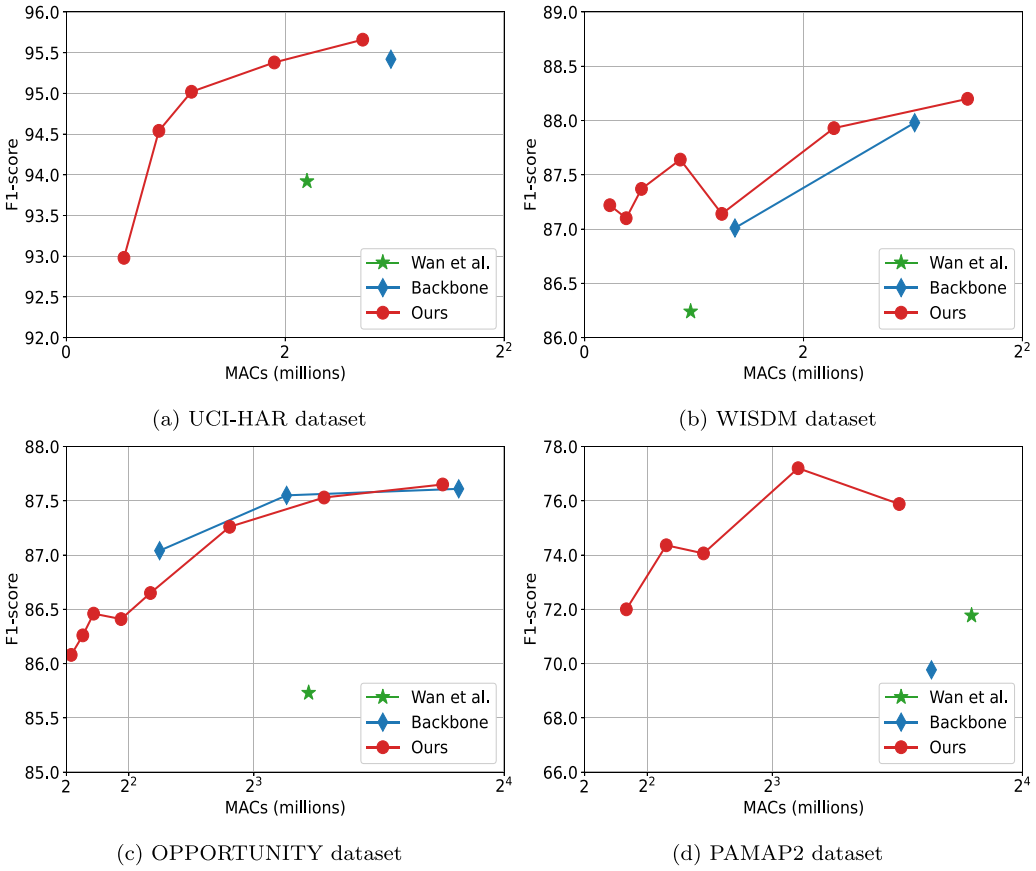
(a) UCI-HAR dataset

(b) WISDM dataset

(c) OPPORTUNITY dataset

(d) PAMAP2 dataset

Fig. 7. Comparison among three networks with regard to the impact of $\mathbb{N}$ on the number of MACs.



(a) UCI-HAR dataset (b) WISDM dataset

(c) OPPORTUNITY dataset (d) PAMAP2 dataset

Fig. 8. Tradeoff between the MACs and F1-score for four HAR datasets.

Table 6
Ablation study of GTSNet for four HAR dataset; P: Permutation within each channel group, S: Shuffle across channel groups.

UCI-HAR	MACs	Model size	F1 score
GTSNet	2.996M	0.078M	0.957
GTSNet without P	2.648M	0.064M	0.952
GTSNet without P & S	2.648M	0.064M	0.948▼
GTSNet with StandardConv	11.220M	0.315M	0.954
WISDM	**MACs**	**Model size**	**F1 score**
GTSNet	1.388M	0.078M	0.886
GTSNet without P	1.225M	0.064M	0.863▼
GTSNet without P & S	1.225M	0.064M	0.857▼
GTSNet with StandardConv	5.242M	0.315M	0.884
OPPORTUNITY	**MACs**	**Model size**	**F1 score**
GTSNet	5.022M	0.092M	0.874
GTSNet without P	4.619M	0.077M	0.868
GTSNet without P & S	4.619M	0.077M	0.867
GTSNet with StandardConv	14.550M	0.327M	0.876
PAMAP2	**MACs**	**Model size**	**F1 score**
GTSNet	13.210M	0.084M	0.762
GTSNet without P	11.817M	0.069M	0.713▼
GTSNet without P & S	11.817M	0.069M	0.732▼
GTSNet with StandardConv	46.107M	0.319M	0.734▼

Table 7
Guidance for $\mathbb{N}$ and G in GTSNet.

TTC	$\mathbb{N} \rightarrow G$							
	32	48	64	96	128	192	256	320
$\mathcal{O}(\mathbb{T}L\mathbb{N}\log(\mathbb{N}))$	↓	↓	↓	↓	↓	↓	↓	↓
	8	12	16	16	32	32	32	64
	30	48	63	96	126	192	255	321
$\mathcal{O}(\mathbb{T}L\mathbb{N})$	↓	↓	↓	↓	↓	↓	↓	↓
	10	16	21	32	42	64	85	107

shown in Table 6, the GTSConv significantly reduces computations and model size of the network. In addition, the results indicate that the permutation and shuffle help preserve accuracy.

6. Discussion

In real-world HAR, it is impractical to achieve a real-time response from a network with $\mathcal{O}(\mathbb{N}^2)$ in various applications. For example, some healthcare systems may require accurate and immediate recognition of an elderly person falling with only a small sensor device. To realize these applications, GTSNet can be used by setting TTC to $\mathcal{O}(\mathbb{T}L\mathbb{N})$ of Corollary 1.3. Given the TTC, the $\mathbb{N}$ of GTSNet increases according to Table 7 until the computational cost of the network reaches the computational budget for the target application.

Meanwhile, $\mathbb{T}$ and L are given and fixed by the sampling frequency of the sensors and the original network, respectively. Commonly, $L \ll \mathbb{T}$ and $L \ll \mathbb{N}$; thus, the search of L has a negligible effect on the efficiency. For the search of $\mathbb{T}$, our framework can be easily integrated with preprocessing methods, including datum-wise frequency selection (Cheng et al., 2018).

In addition, another important factor in real-world applications is a memory access cost (MACC). If the cache in the device is large enough to store intermediate feature signals and model parameters, MACC of StandardConv is $T(M+N)+D_k MN$ (Ma et al., 2018); herein, the two terms are related to the memory access for the input/output signals and kernel parameters, respectively. Therefore, the time complexity for the MACC in the backbone network, described in Section 4.1, is $\mathcal{O}(\mathbb{N}L(\mathbb{T}+\mathbb{D}_k\mathbb{N}))$. As the number of parameters of the GTS module is $2*MN/G$, our GTSNet has the time complexity of $\mathcal{O}(\mathbb{N}L(\mathbb{T}+\mathbb{N}/G))$ for the

Table 8
Comparison results with large-scale CNNs on the four HAR datasets.

UCI-HAR	MACs	Model size	F1 score	Inference time
GTSNet	2.996M	0.078M	0.957	3.84 ms
ViT	341.025M	37.908M	0.922▼	12.08 ms
ResNet-50	101.030M	15.969M	0.937▼	6.53 ms
VGG-16	104.364M	36.407M	0.928▼	6.78 ms
WISDM	MACs	Model size	F1 score	Inference time
GTSNet	1.388M	0.078M	0.886	3.62 ms
ViT	189.204M	37.855M	0.858▼	9.97 ms
ResNet-50	50.298M	15.967M	0.868▼	5.80 ms
VGG-16	62.852M	36.406M	0.891	6.49 ms
OPPORTUNITY	MACs	Model size	F1 score	Inference time
GTSNet	5.022M	0.092M	0.874	4.31 ms
ViT	433.282M	39.561M	0.885△	13.46 ms
ResNet-50	128.600M	16.041M	0.883△	8.73 ms
VGG-16	117.406M	36.476M	0.877	7.77 ms
PAMAP2	MACs	Model size	F1 score	Inference time
GTSNet	13.210M	0.084M	0.762	4.44 ms
ViT	1.264G	38.330M	0.769△	22.74 ms
ResNet-50	406.968M	16.005M	0.777△	15.63 ms
VGG-16	325.505M	36.461M	0.734▼	9.89 ms

MACC in accordance with Lemma 3. Because $\mathbb{D}_k\mathbb{N} > \mathbb{N}/G$, our GTSNet has a more efficient MACC than the network with StandardConv.

Furthermore, we also consider a disk access cost (DACC). If the cache in the device is not large enough, the disk access can be required whenever calculating the output of each layer. As the input of each layer is removed after calculating its output in an inference step, the disk access cost depends on the parameter size of each layer. As shown in Tables 2–5, our GTSNet has the lowest model size, resulting in the lowest DACC. In conclusion, our GTSNet is more efficient than other models on resource-limited devices with low memory or disk bandwidth.

Lastly, we need to examine whether the accuracy of our GTSNet is large enough for use in various real-world applications. To this end, we compared the GTSNet with large-scale networks (ViT Dosovitskiy et al., 2021 and ResNet50 He et al., 2016, and VGG16 Simonyan and Zisserman, 2015). As shown in Table 8, GTSNet outperforms ViT and ResNet-50 on WISDM and UCI-HAR but not OPPORTUNITY and PAMAP2 datasets. This limitation indicates that GTSNet requires more sophisticated network architectures to classify complex activities correctly. In addition, Table 8 shows the inference time of the four networks, measured in AMD Ryzen 7 5800X 8-Core Processor. It indicates our GTSNet can achieve the computation budget even in a single CPU setting. Although they show minor differences compared to the MACs, these gaps can get wider and wider on devices with low memory bandwidth due to the tiny size of GTSNet.

7. Conclusion

In this paper, we proposed a novel deep-learning design framework that reconstructs the network architecture without requiring exhaustive or labor-intensive manual redesign for real-time HAR from wearable sensors. Specifically, we formalized the theoretical computation cost of the network architecture by using the time complexity as the upper bound of the cost. After that, we introduced the GTSNet, which allows its architecture to be flexibly modified by predefining the desired computation budget. Our experiments showed that our GTSNet possesses the superior tradeoff between accuracy and computational cost, resulting in the highest accuracy under the given computational budget. In addition, the ablation study showed that the proposed framework could derive an efficient architecture while preserving the HAR accuracy.

Future studies can be conducted to overcome the limitations of the proposed framework. First, the network derived from our framework should be trained from scratch, abandoning the pretrained weights of the backbone network. To address this issue, various transfer learning techniques can be used. Second, our framework does not consider runtime optimizations of model graphs. For example, TensorFlow Lite merges multiple operators into one fused operator to boost the inference time. In this regard, future studies may further improve the tradeoff between accuracy and efficiency by formulating theoretical computation budgets more sophisticatedly.

CRediT authorship contribution statement

Jaegyun Park: Conceptualization, Methodology, Software, Formal analysis, Writing – original draft, Visualization. **Won-Seon Lim:** Methodology, Software, Validation, Investigation. **Dae-Won Kim:** Conceptualization, Methodology, Writing – review & editing, Supervision. **Jaesung Lee:** Methodology, Formal analysis, Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Jaesung Lee reports financial support was provided by Institute for Information Communication Technology Planning and Evaluation. Dae-Won Kim reports financial support was provided by National Research Foundation of Korea.

Data availability

Data will be made available on request.

Acknowledgments

This research was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (2021-0-01341, Artificial Intelligence Graduate School Program (Chung-Ang University)) and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2023R1A2C1006745).

References

Anguita, D., Ghio, A., Oneto, L., Parra, X., Reyes-Ortiz, J., 2013. 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, pp. 437–442.

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

Chavarriaga, R., Sagha, H., Calatroni, A., Digumarti, S.T., Tröster, G., Millán, J.d.R., Roggen, D., 2013. The opportunity challenge: A benchmark database for on-body sensor-based activity recognition. Pattern Recognit. Lett. 34 (15), 2033–2042.

Chen, K., Zhang, D., Yao, L., Guo, B., Yu, Z., Liu, Y., 2021. Deep learning for sensor-based human activity recognition: Overview, challenges, and opportunities. ACM Comput. Surv. 54 (4), 1–40.

Cheng, W., Erfani, S., Zhang, R., Kotagiri, R., 2018. Learning datum-wise sampling frequency for energy-efficient human activity recognition. In: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32.

Cheng, X., Zhang, L., Tang, Y., Liu, Y., Wu, H., He, J., 2022. Real-time human activity recognition using conditionally parametrized convolutions on mobile and wearable devices. IEEE Sens. J..

Chollet, F., 2017. Xception: Deep learning with depthwise separable convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1251–1258.

Dang, L.M., Min, K., Wang, H., Piran, M.J., Lee, C.H., Moon, H., 2020. Sensor-based and vision-based human activity recognition: A comprehensive survey. Pattern Recognit. 108, 107561.

Dempster, A., Petitjean, F., Webb, G.I., 2020. ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels. Data Min. Knowl. Discov. 34 (5), 1454–1495.

Ding, Y., Li, K., Liu, C., Tang, Z., Li, K., 2021. Budget-constrained service allocation optimization for mobile edge computing. IEEE Trans. Serv. Comput..

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N., 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In: 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021.

Fawaz, H.I., Forestier, G., Weber, J., Idoumghar, L., Muller, P.-A., 2019. Deep learning for time series classification: a review. Data Min. Knowl. Discov. 33 (4), 917–963.

Franceschi, J.-Y., Dieuleveut, A., Jaggi, M., 2019. Unsupervised scalable representation learning for multivariate time series. Adv. Neural Inf. Process. Syst. 32.

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

Gehring, J., Auli, M., Grangier, D., Yarats, D., Dauphin, Y.N., 2017. Convolutional sequence to sequence learning. In: International Conference on Machine Learning. PMLR, pp. 1243–1252.

Ghadami, N., Gheibi, M., Kian, Z., Faramarz, M.G., Naghedi, R., Eftekhari, M., Fathollahi-Fard, A.M., Dulebenets, M.A., Tian, G., 2021. Implementation of solar energy in smart cities using an integration of artificial neural network, photovoltaic system and classical Delphi methods. Sustainable Cities Soc. 74, 103149.

Gholizadeh, H., Fathollahi-Fard, A.M., Fazlollahtabar, H., Charles, V., 2022. Fuzzy data-driven scenario-based robust data envelopment analysis for prediction and optimisation of an electrical discharge machine's parameters. Expert Syst. Appl. 193, 116419.

Graves, A., Schmidhuber, J., 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Netw. 18 (5–6), 602–610.

Guan, Y., Plötz, T., 2017. Ensembles of deep lstm learners for activity recognition using wearables. Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 1 (2), 1–28.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778.

He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., Li, M., 2019. Bag of tricks for image classification with convolutional neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 558–567.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9 (8), 1735–1780.

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H., 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

Huang, J., Lin, S., Wang, N., Dai, G., Xie, Y., Zhou, J., 2019. TSE-CNN: A two-stage end-to-end CNN for human activity recognition. IEEE J. Biomed. Health Inf. 24 (1), 292–299.

Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K., 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5 MB model size. arXiv preprint arXiv:1602.07360.

Ignatov, A., 2018. Real-time human activity recognition from accelerometer data using Convolutional Neural Networks. Appl. Soft Comput. 62, 915–922.

Jeon, Y., Kim, J., 2018. Constructing fast network through deconstruction of convolution. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems. pp. 5955–5965.

Karim, F., Majumdar, S., Darabi, H., Chen, S., 2017. LSTM fully convolutional networks for time series classification. IEEE Access 6, 1662–1669.

Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Koli, R.R., Bagban, T.I., 2020. Human action recognition using deep neural networks. In: 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4). IEEE, pp. 376–380.

Kwapisz, J.R., Weiss, G.M., Moore, S.A., 2011. Activity recognition using cell phone accelerometers. ACM SigKDD Explor. Newsl. 12 (2), 74–82.

Li, H., Shrestha, A., Heidari, H., Le Kernec, J., Fioranelli, F., 2019. Bi-LSTM network for multimodal continuous human activity recognition and fall detection. IEEE Sens. J. 20 (3), 1191–1201.

Lv, M., Xu, W., Chen, T., 2019. A hybrid deep convolutional and recurrent neural network for complex activity recognition using multimodal sensors. Neurocomputing 362, 33–40.

Ma, N., Zhang, X., Zheng, H.-T., Sun, J., 2018. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 116–131.

Mikolov, T., et al., 2012. Statistical language models based on neural networks. Present. Google Mt. View 2nd April 80 (26).

Murad, A., Pyun, J.-Y., 2017. Deep recurrent neural networks for human activity recognition. Sensors 17 (11), 2556.

Oluwalade, B., Neela, S., Wawira, J., Adejumo, T., Purkayastha, S., 2021. Human activity recognition using deep learning models on smartphones and smartwatches sensor data. arXiv preprint arXiv:2103.03836.

Ordóñez, F.J., Roggen, D., 2016. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. Sensors 16 (1), 115.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al., 2019. Pytorch: An imperative style, high-performance deep learning library. Adv. Neural Inf. Process. Syst. 32, 8026–8037.

Peng, L., Chen, L., Ye, Z., Zhang, Y., 2018. Aroma: A deep multi-task learning based simple and complex human activity recognition method using wearable sensors. Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 2 (2), 1–16.

Qi, J., Yang, P., Waraich, A., Deng, Z., Zhao, Y., Yang, Y., 2018. Examining sensor-based physical activity recognition and monitoring for healthcare using internet of things: A systematic review. J. Biomed. Inform. 87, 138–153.

Ravi, D., Wong, C., Lo, B., Yang, G.-Z., 2016. A deep learning approach to on-node sensor data analytics for mobile or wearable devices. IEEE J. Biomed. Health Inf. 21 (1), 56–64.

Reiss, A., Stricker, D., 2012. Introducing a new benchmarked dataset for activity monitoring. In: 2012 16th International Symposium on Wearable Computers. IEEE, pp. 108–109.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.-C., 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4510–4520.

Shahsavar, M.M., Akrami, M., Gheibi, M., Kavianpour, B., Fathollahi-Fard, A.M., Behzadian, K., 2021. Constructing a smart framework for supplying the biogas energy in green buildings using an integration of response surface methodology, artificial intelligence and petri net modelling. Energy Convers. Manage. 248, 114794.

Shifaz, A., Pelletier, C., Petitjean, F., Webb, G.I., 2020. TS-CHIEF: a scalable and accurate forest algorithm for time series classification. Data Min. Knowl. Discov. 34 (3), 742–775.

Shoaib, M., Bosch, S., Incel, O.D., Scholten, H., Havinga, P.J., 2015. A survey of online activity recognition using mobile phones. Sensors 15 (1), 2059–2085.

Simonyan, K., Zisserman, A., 2015. Very deep convolutional networks for large-scale image recognition. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.

Wan, S., Qi, L., Xu, X., Tong, C., Gu, Z., 2020. Deep learning models for real-time human activity recognition with smartphones. Mob. Netw. Appl. 25 (2), 743–755.

Wang, J., Chen, Y., Hao, S., Peng, X., Hu, L., 2019. Deep learning for sensor-based activity recognition: A survey. Pattern Recognit. Lett. 119, 3–11.

Wang, J., Wang, Z., Li, J., Wu, J., 2018. Multilevel wavelet decomposition network for interpretable time series analysis. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 2437–2446.

Wang, Z., Yan, W., Oates, T., 2017. Time series classification from scratch with deep neural networks: A strong baseline. In: 2017 International Joint Conference on Neural Networks (IJCNN). IEEE, pp. 1578–1585.

Wu, B., Wan, A., Yue, X., Jin, P., Zhao, S., Golmant, N., Gholaminejad, A., Gonzalez, J., Keutzer, K., 2018. Shift: A zero flop, zero parameter alternative to spatial convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9127–9135.

Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K., 2017. Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1492–1500.

Xu, C., Chai, D., He, J., Zhang, X., Duan, S., 2019a. InnoHAR: A deep neural network for complex human activity recognition. IEEE Access 7, 9893–9902.

Xu, M., Qian, F., Zhu, M., Huang, F., Pushp, S., Liu, X., 2019b. Deepwear: Adaptive local offloading for on-wearable deep learning. IEEE Trans. Mob. Comput. 19 (2), 314–330.

Yu, J., Yang, L., Xu, N., Yang, J., Huang, T., 2018. Slimmable neural networks. In: International Conference on Learning Representations.

Zebin, T., Scully, P.J., Peek, N., Casson, A.J., Ozanyan, K.B., 2019. Design and implementation of a convolutional neural network on an edge computing smartphone for human activity recognition. IEEE Access 7, 133509–133520.

Zhang, X., Zhou, X., Lin, M., Sun, J., 2018. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6848–6856.