scientific reports

Check for updates

OPEN

Efficient human activity recognition on edge devices using DeepConv LSTM architectures

Haotian Zhou[1], Xiujun Zhang[2], Yu Feng[1], Tongda Zhang[1] & Lijuan Xiong[3]✉

Driven by the rapid development of the Internet of Things (IoT), deploying deep learning models on resource-constrained hardware has become an increasingly critical challenge, which has propelled the emergence of TinyML as a viable solution. This study aims to deploy lightweight deep learning models for human activity recognition (HAR) using TinyML on edge devices. We designed and evaluated three models: a 2D Convolutional Neural Network (2D CNN), a 1D Convolutional Neural Network (1D CNN), and a DeepConv LSTM. Among these, the DeepConv LSTM outperformed existing lightweight models by effectively capturing both spatial and temporal features, achieving an accuracy of 98.24% and an F1 score of 98.23%. After performing full integer quantization on the best model, its size was reduced from 513.23 KB to 136.51 KB and was successfully deployed on the Arduino Nano 33 BLE Sense Rev2 using the Edge Impulse platform. The device's memory usage was 29.1 KB, flash usage was 189.6 KB, and the model's average inference time was 21 milliseconds, requiring approximately 0.01395 GOP, with a computational performance of around 0.664 GOPS. Even after quantization, the model maintained an accuracy of 97% and an F1 score of 97%, ensuring efficient utilization of computational resources. This deployment highlights the potential of TinyML in achieving low-latency and efficient HAR systems, making it suitable for real-time human activity recognition applications.

Keywords TinyML, IoT, Deep Learning, Edge Computing, Model Quantization, Human Activity Recognition

In recent years, the integration of IoT, AI, and cloud computing has spurred considerable innovation across various domains.By 2030, it is projected that there will be more than 30 billion IoT devices globally[1]. However, these applications often require significant computing resources, including memory, CPU, and network bandwidth. As data continues to grow, traditional cloud computing solutions may struggle to keep up[2]. By 2035, autonomous vehicles are expected to account for approximately 66% of total car sales in China[3], presenting significant challenges for traditional cloud-based methods. In particular, data transmission errors or malicious attacks could lead to incorrect decisions, causing serious social and economic consequences[4]. Therefore, exploring more sustainable distributed computing models, such as edge computing, has become increasingly important[5]. Edge computing reduces dependence on centralized cloud services by processing data on local devices, thereby lowering latency and improving real-time processing capabilities[6].

Human Activity Recognition (HAR) is a classification problem that involves predicting activities such as jogging, lying down, and standing based on data collected from individual users[7]. HAR has vast potential applications, including smart homes, healthcare, and other automated domains reliant on human behavior analysis. For example, HAR can be used to monitor the health status of elderly individuals[8] and assist people with disabilities[9].

HAR is primarily achieved through wearable sensors[10] and cameras[11]. In comparison, sensor-based solutions offer higher efficiency and better privacy protection, and they are not affected by environmental factors such as lighting[12].Moreover, the significantly lower cost of sensors compared to their counterparts has led to rapid research advancements in this field[13]. Today, sensor-based HAR applications have expanded into various fields, including healthcare systems[14,15], vehicle travel time prediction[16], and context-aware systems[17].

Initially, HAR relied primarily on traditional machine learning techniques, achieving significant results[18]. For example,[19] analyzed various machine learning models, including Decision Tree, Random Forest, and XGBoost, for HAR on wearable devices, achieving state-of-the-art accuracy on the Human Activity Recognition with Smartphones dataset. XGBoost, in particular, achieved an accuracy of 99.52%, demonstrating the potential

[1]School of Computer Science, Chengdu University, Chengdu 610106, China. [2]School of Artificial Intelligence, Chengdu Polytechnic, Chengdu 610041, China. [3]School of Fine Arts and Design, Chengdu University, Chengdu 610106, China. ✉email: xionglijuan@cdu.edu.cn

of machine learning models in HAR tasks. However, these methods typically require substantial computational resources and considerable manual intervention by domain experts, making it difficult to adapt to new environments. Consequently, research has gradually shifted from traditional machine learning to deep learning, opening up broader possibilities.Due to its hierarchical structure, deep learning enables models to learn from raw, unprocessed data.

In recent human activity recognition (HAR) research, reference[20] proposed a stacked LSTM network, achieving an accuracy of 93.13% on the UCI dataset. However, this model relies solely on LSTM units and fails to fully capture temporal features, leaving room for improvement in accuracy. Reference[21] introduced a CNN-GRU model, which achieved an accuracy of 96.54% on the WISDM dataset. However, its complex structure increases computational resource demands, making it difficult to deploy in resource-constrained environments. Reference[22] designed a hybrid CNN and BiLSTM model, which captures bidirectional dependencies through bidirectional LSTM, achieving an accuracy of 96.05% on the WISDM dataset. Nevertheless, the model's multi-branch architecture and extensive use of convolutional and pooling layers significantly increase computational overhead, limiting its applicability on edge devices. Reference[23] proposed a CNN-LSTM model incorporating a self-attention mechanism, achieving an accuracy of 99.93% on its proprietary dataset and 93.11% on the UCI-HAR dataset. Although the attention mechanism enhances the model's classification capabilities, it also significantly increases storage and computational requirements, making it challenging to deploy in resource-constrained environments. Therefore, how to reduce computational demands while maintaining high accuracy and enabling efficient deployment on IoT devices remains a critical challenge.

To address these challenges, this study focuses on designing lightweight deep learning models that can be efficiently deployed on resource-constrained devices. In[24], the authors explored four architectures (CNN, LSTM, ConvLSTM2D, and CNN-LSTM), among which the CNN-LSTM model achieved the highest accuracy of 97.68% on the WISDM dataset. However,[25] used CNN-LSTM and CNN-GRU models, but none exceeded 98% accuracy. Additionally, in[26], the authors compared 1D CNN and 2D CNN and ultimately selected 1D CNN for training, achieving an accuracy of 97.8%.In[27], the authors proposed a neural architecture search (NAS) method based on a latency lookup table (LUT) to optimize HAR models for mobile devices. Their approach significantly reduced search time and improved inference efficiency by tailoring models to specific hardware constraints. However, on the WISDM dataset, their model achieved an F1 score of only 88%, indicating room for improvement in balancing accuracy and efficiency.Although these methods have achieved promising results, there is still room for improvement in model performance. Therefore, we selected three primary architectures for evaluation: 1D CNN, 2D CNN, and DeepConv LSTM. 1D CNN was chosen for its efficiency in extracting temporal features from sequential sensor data, while 2D CNN was selected for its capability to capture spatial features. DeepConv LSTM integrates convolutional layers with LSTM networks, allowing it to capture both spatial and temporal dependencies, making it more suitable for recognizing complex human activities.

With the rapid development of IoT devices, they are now capable of supporting more advanced algorithms at the edge computing layer[28]. This progress reduces the reliance on cloud computing, thereby improving response times, lowering latency, and reducing bandwidth consumption. Consequently, the concept of integrating artificial intelligence with IoT devices has emerged, combining AI technology into IoT systems[29]. Traditionally, deep learning algorithms primarily relied on cloud computing for processing. However, in real-time analysis scenarios, high latency can occur, affecting service quality. In contrast, TinyML[30] focuses on developing models that can be deployed on IoT devices, enabling these devices to make decisions at the point of data generation. TinyML has now become a new trend[31] and is being widely applied across various domains.

For example, in[32], LSTM autoencoders were used for unsupervised anomaly detection of urban noise, achieving an accuracy of 99.99%. After quantization, the model was successfully deployed on microcontroller units (MCUs) with an average inference time of approximately 4 milliseconds. In[33], TinyML sensors predicted the shelf life of date fruits, with both training and deployment conducted on the Edge Impulse platform, significantly reducing computational resource requirements and energy consumption, with all samples achieving over 93% accuracy. In[34], the authors proposed an anomaly detection system based on the Isolation Forest algorithm, capable of detecting anomalies within 16 milliseconds, demonstrating the low-latency and adaptability of TinyML in extreme industrial environments. In[35], a binarized neural network was deployed on a Raspberry Pi for low-latency human activity recognition, utilizing 721KB of memory and achieving 98.2% accuracy. Therefore, the combination of TinyML and human activity recognition (HAR) offers advantages such as low power consumption, enhanced security, and fast response times, addressing hardware and connectivity challenges while improving system efficiency[36].

The main contributions of this study are:

1. We designed three Human Activity Recognition (HAR) models and identified DeepConv LSTM as the optimal model, achieving an accuracy of 98.24%. Compared with existing lightweight HAR models, our proposed model demonstrates superior performance.
2. The model was fully quantized and deployed on the Arduino Nano 33 BLE Sense Rev2, addressing the gap in previous studies where lightweight models were not deployed on resource-constrained devices.
3. We analyzed the feasibility of deploying the model to the Arduino Nano 33 BLE Sense Rev2, including model inference time, flash memory, RAM usage, and power consumption, ensuring efficient execution on resource-constrained devices.

The remainder of this paper is structured as follows: Section "Methods" Presents the preprocessing methods and provides a detailed description of the three model architectures. Section "Results" evaluates our models on the WISDM dataset and analyzed the results to identify the best-performing model for deployment. Section "Discussion" summarizes the findings and proposed directions for future work.

Methods

Data analysis

The dataset used in this study was obtained from the WISDM Lab at the Department of Computer and Information Science, Fordham University, located in the Bronx, New York[37]. During data collection, 36 volunteers placed an Android smartphone in their front pants pocket and performed a series of common human activities. The accelerometer data was recorded at a frequency of 20 Hz, meaning one data point was captured every 50 milliseconds, equating to 20 samples per second. Figure 1a illustrates the distribution of samples across different activity categories. Notably, the "standing" and "sitting" activities have the fewest samples, as these static activities can be accurately recognized without requiring a large amount of data.

As shown in Fig. 1b, not all volunteers participated in every activity. Additionally, the duration for which each volunteer performed each activity varied. However, since the data volume for each activity is sufficiently large, these variations have little impact on subsequent processing and analysis. This diversity in the data enhances the model's generalization capability, enabling more accurate recognition of different human activities.

Ethical approval and informed consent

The data used in our study are publicly available, and ethical approval and informed consent were obtained in the original study, in which Jennifer R. Kwapisz and colleagues obtained approval from the Fordham University Institutional Review Board (IRB)[37].

Data pre-processing

To ensure data usability, comprehensive preprocessing was performed prior to training. Initially, rows containing null values and those with zero timestamps were removed, and the data was sorted by user and timestamp. As shown in Fig. 1a, the dataset exhibits a certain degree of imbalance. However, unlike the approach taken in reference[38], we did not employ SMOTE or analogous techniques to equilibrate the dataset. This decision

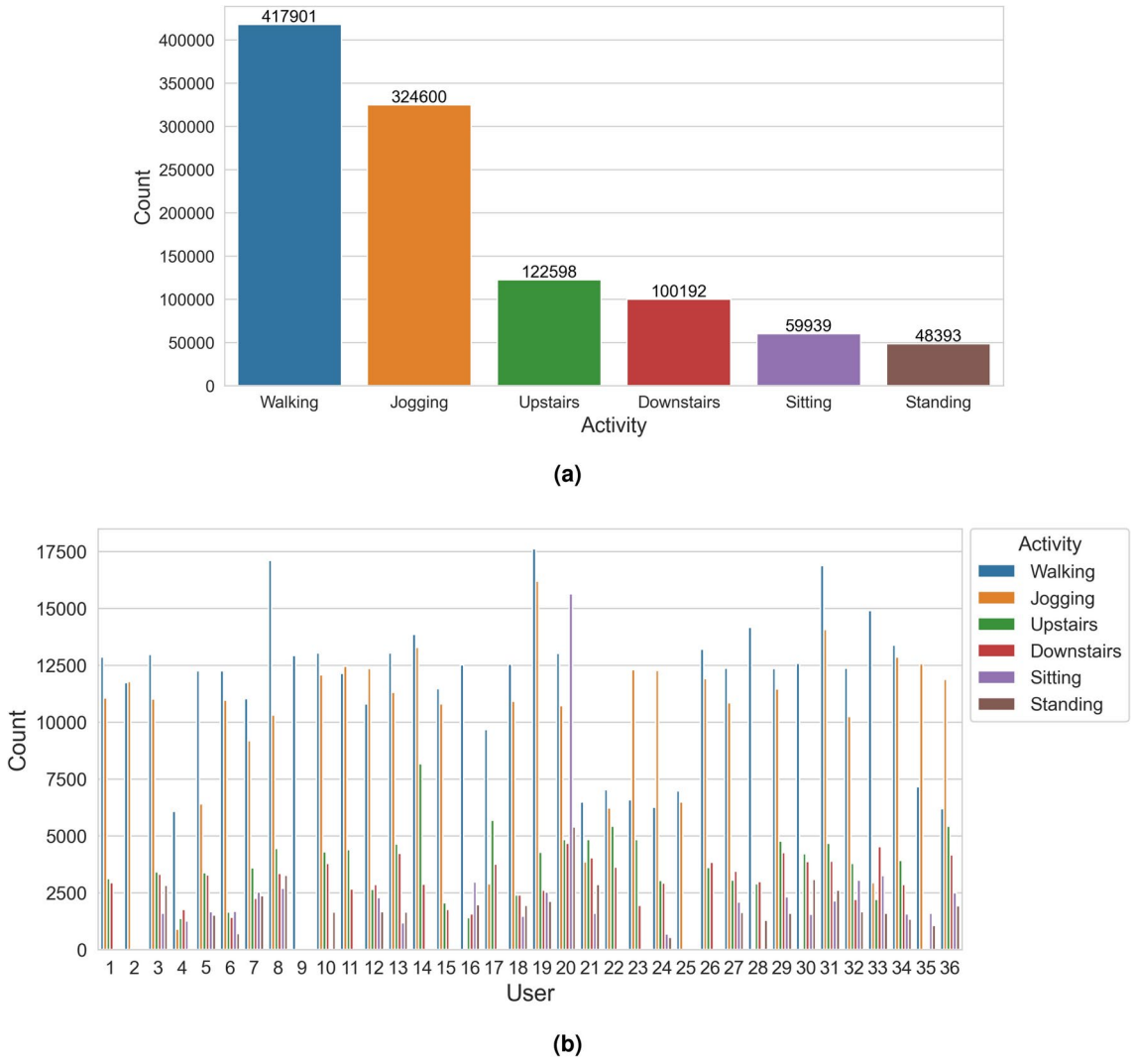


Fig. 1. (**a**) Number of samples by activity (**b**) Number of samples by Users.

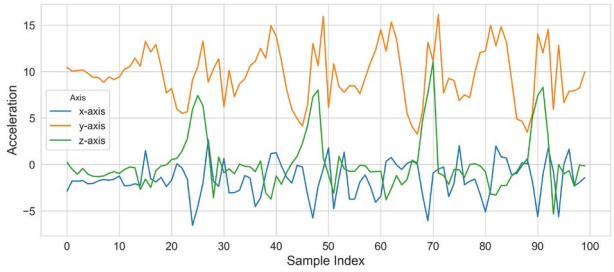


Fig. 2. Activities | Walking.

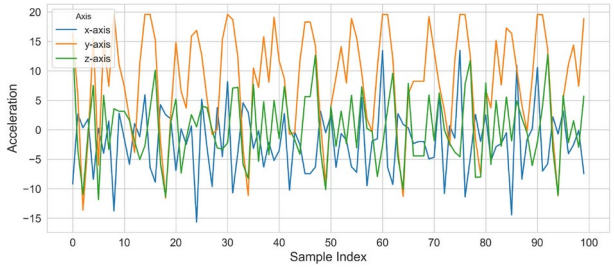


Fig. 3. Activities | Jogging.

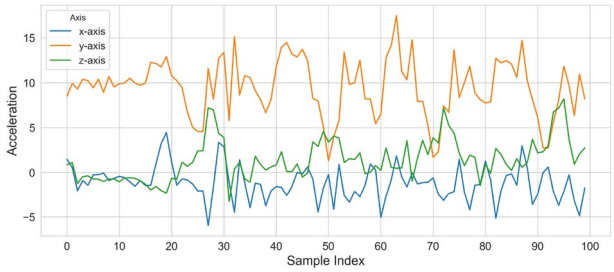


Fig. 4. Activities | Upstairs.

was primarily made to preserve the original data distribution and avoid potential information redundancy or distribution shifts caused by oversampling. Moreover, oversampling may reduce the model's generalization ability for certain classes, ultimately affecting overall recognition performance. Meanwhile,[18,24–26] also did not apply data balancing techniques. Despite the class imbalance, the models in these studies still achieved high classification accuracy across all categories and did not exhibit significant bias toward majority classes, indicating that data imbalance had a minimal impact on recognition performance.

Since raw accelerometer data cannot be directly applied to classification algorithms, it was necessary to convert the raw data into usable samples. The window size is crucial in determining whether the model can effectively capture features; too large or too small a window may result in insufficient feature extraction or a reduction in available data points for training.According to the literature[39,40], short windows (1-2 seconds) are suitable for detecting postures and simple activities, while slightly longer windows (4-5 seconds) are more effective in distinguishing various activities. For recognizing complex movement patterns, longer windows (10 seconds or more) perform best. Research[41] indicates that among window sizes of 32, 64, and 128, a window size of 128 yielded the best results. Therefore, this study validates different window sizes and selects the most suitable window size based on the accuracy achieved by the models. Figures 2, 3, 4, 5, 6 and 7 show the temporal changes of the first 100 samples for each activity in user 36. Figures 2 and 3 demonstrate periodic acceleration changes, and this periodicity can more effectively capture spatial features. As seen in Figures 4 and 5, the upstairs and downstairs activities exhibit similar acceleration patterns, making it more challenging to distinguish between them. Misclassification between these two activities is common in HAR models due to their overlapping motion characteristics. In contrast, Figures 6 and 7 show significantly reduced acceleration variations for sitting and standing activities, making them relatively easier to classify. To maintain data continuity, a sliding window with 50% overlap was used[39], rather than discrete windows. Finally, the processed dataset was split, with 33% reserved for testing. As shown in Fig. 8, each activity category was proportionally divided to ensure that the test set remains representative and provides sufficient samples for model evaluation. Additionally, to preserve data authenticity,

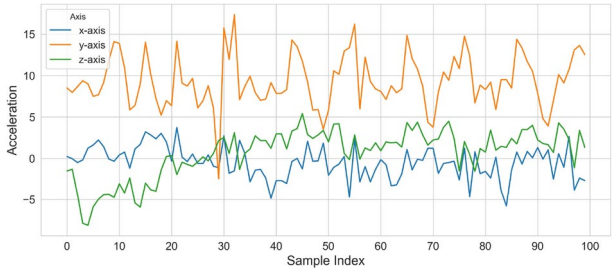


Fig. 5. Activities | Downstairs.

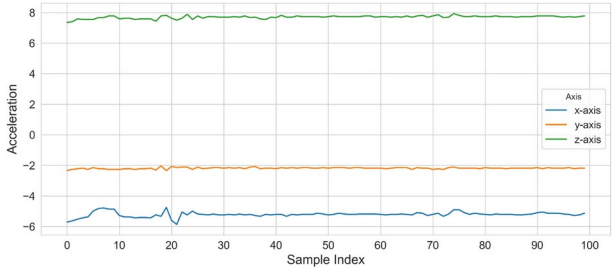


Fig. 6. Activities | Sitting.

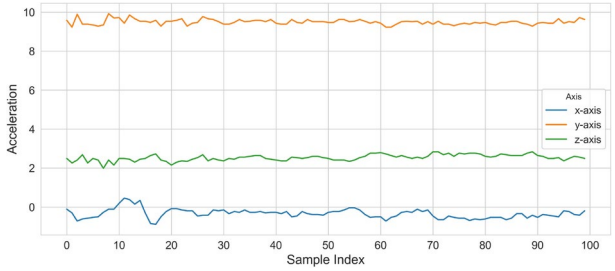


Fig. 7. Activities | Standing.

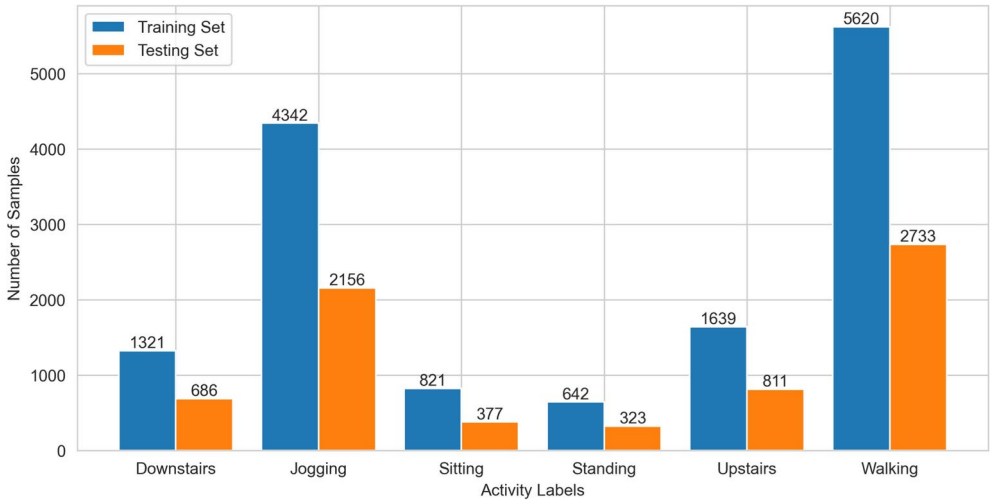


Fig. 8. Distribution of Activities in Training and Testing Sets.

no oversampling or synthetic augmentation techniques were applied, allowing for a more accurate assessment of the model's real-world performance. The data partitioning strategy ensures a balanced distribution of activity classes in both the training and test sets, reducing potential bias and enhancing the model's generalization ability.

Long short-term memory (LSTM)

LSTM is a special type of recurrent neural network introduced by Hochreiter and Schmidhuber in 1997 to address the gradient vanishing and exploding problems often encountered by RNNs when processing long sequences of data[42]. LSTM networks are capable of capturing critical information in time-series data and effectively prevent the gradient vanishing issue. The core of LSTM lies in the introduction of a mechanism called the cell state, which functions like a conveyor belt, allowing information to propagate linearly through the sequence and remain relatively unchanged, thus preventing information loss during transmission.

As shown in Fig. 9, the core components of LSTM include the forget gate, the input gate, and the output gate. The forget gate filters out unimportant content to prevent irrelevant information from causing interference. The input gate updates the cell state by combining the current input with the hidden state from the previous step. The output gate determines which parts of the cell state will be passed on to the next LSTM unit.

The working process of LSTM can be represented by the following equations:

The forget gate is controlled by a sigmoid function. Its output ranges between 0 and 1(W_f is the weight matrix of the forget gate, and b_f is the bias term of the forget gate):

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{1}$$

The input gate generates activation values through a sigmoid function, and then a tanh function is used to create the candidate cell state $\tilde{C}t$:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{3}$$

C t is obtained by weighted summation of the results from equations (1), (2), and (3):

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \tag{4}$$

The output gate calculates the activation values of the output gate using the sigmoid function, and then combines the current cell state with the tanh function (4) to obtain the new hidden state h_t:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{5}$$

$$h_t = o_t \cdot \tanh(C_t) \tag{6}$$

Model

In this study, we designed three models: the 1D CNN model, the 2D CNN model, and the DeepConv LSTM model. To further enhance model performance and training stability, we conducted systematic empirical testing on key hyperparameters, including learning rate, batch size, convolutional filters, LSTM units, and dropout rates. We also incorporated training optimization techniques, such as ReduceLROnPlateau (to dynamically adjust the learning rate when validation loss plateaus) and EarlyStopping (to prevent overfitting by stopping training when validation loss does not improve for a specified number of epochs). These strategies helped balance model accuracy, generalization, and computational efficiency.All experiments were conducted on a system equipped with an Intel Core i7-9750H CPU, 32GB of RAM, and an NVIDIA GeForce GTX 1650 (4GB) GPU. Each model features a unique architecture and feature extraction capability, aimed at addressing the challenges of real-time human activity recognition on resource-constrained devices. The following sections provide a detailed introduction to the architecture, characteristics, and training strategies of each model, along with the final optimized hyperparameters selected for best performance. Additionally, the complete model code has

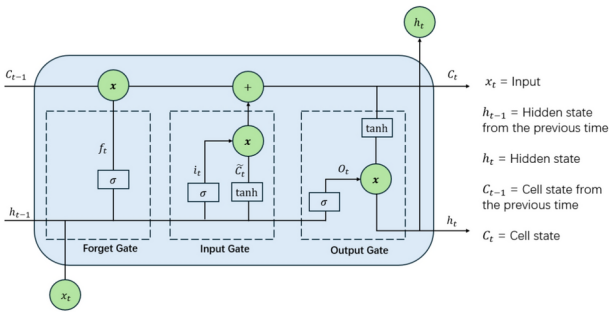


Fig. 9. LSTM architecture.

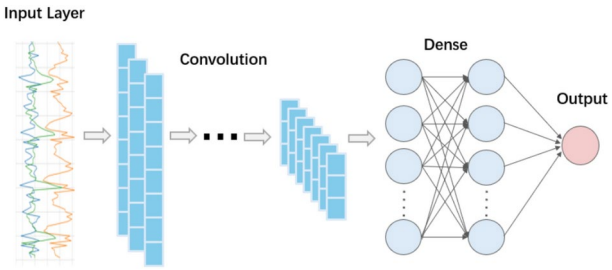


Fig. 10. Basic structure of 1D CNN model.

Layer	Output Shape	Param #
Conv1D	(None, 98, 64)	640
Dropout	(None, 98, 64)	0
Conv1D	(None, 94, 64)	12352
Dropout	(None, 98, 64)	0
Conv1D	(None, 86, 64)	12352
Dropout	(None, 86, 64)	0
Flatten	(None, 5504)	0
Dense	(None, 64)	352320
Dense	(None, 6)	390

Table 1. 1D CNN Model summary. Total params: 378,054. Trainable params: 378,054 Non-trainable params: 0

been uploaded to https://github.com/zsgdszt/haotianzhou/blob/master/ to ensure reproducibility and facilitate further research.

1D CNN model

The 1D Convolutional Neural Network (1D CNN) is commonly used for feature extraction. As depicted in Fig. 10, its basic architecture comprises one-dimensional convolutional layers and fully connected layers. The convolutional layers move along the temporal axis to extract local features from raw time-series data. Finally, the fully connected layers leverage the extracted features for classification, facilitating the recognition of various categories.

In existing research, 1D CNN models have been widely used for time-series classification tasks, achieving significant success. For example, in the study by Cho et al.[43], two distinct 1D CNN models were developed to separately classify dynamic and static activities. A convolution window size of 3 was set to extract features. In the dynamic activity model, a max-pooling layer was added to reduce the number of features and prevent overfitting. In contrast, in the static activity model, three consecutive convolutional layers were used without pooling to preserve more spatial information. Both models employed a high dropout rate of 50% to reduce overfitting, and the classification results were output through a softmax layer, achieving an accuracy of 97.62% on the UCI HAR dataset. Similarly,[44] designed a 1D CNN model for classifying time-series sensor data. This study demonstrated that 1D CNN models can effectively handle raw sensor input data, achieving both feature extraction and classification. Even with fewer parameters, this approach significantly outperformed traditional machine learning methods (e.g., Naive Bayes, SVM, and KNN) in terms of speed and accuracy.

Based on these research findings, a 1D CNN model (Table 1) was designed to improve the accuracy and generalization capability of human activity recognition while ensuring computational efficiency. The model consists of three 1D convolutional layers, each with 64 filters and a kernel size of 3, and employs dilation rates of 1, 2, and 4 to capture multi-scale temporal dependencies without significantly increasing computational complexity. A dropout layer (rate = 0.2) follows each convolutional layer to prevent overfitting while maintaining feature integrity. Compared to the 50% dropout rate used in some previous studies, this lower rate provides a better balance between regularization and feature retention. The feature representations extracted by the convolutional layers are flattened and passed through a fully connected layer (64 units, ReLU activation) before being classified by a softmax layer.The model was trained using categorical cross-entropy loss and optimized with RMSprop, where the initial learning rate was set to 0.001. The training process was conducted with a batch size of 64 for 50 epochs. To further improve model stability and prevent overfitting, ReduceLROnPlateau was employed to reduce the learning rate by a factor of 0.5 when validation loss stagnated for five consecutive epochs, while EarlyStopping was applied to terminate training if validation loss did not improve for ten consecutive epochs, restoring the best-performing model weights.

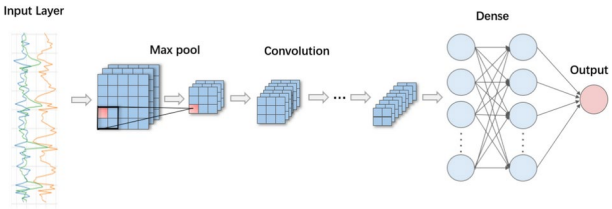


Fig. 11. Basic structure of 2D CNN model.

Layer	Output Shape	Param #
Reshape	(None, 100, 3, 1)	0
Conv2D	(None, 100, 3, 32)	128
Conv2D	(None, 100, 3, 32)	3104
MaxPooling2D	(None, 50, 3, 32)	0
Conv2D	(None, 50, 3, 64)	6208
Conv2D	(None, 50, 3, 64)	12352
MaxPooling2D	(None, 25, 3, 64)	0
Flatten	(None, 4800)	0
Dense	(None, 128)	614528
Dropout	(None, 128)	0
Dense	(None, 6)	774

Table 2. 2D CNN Model summary. Total params: 637,094 Trainable params: 637,094 Non-trainable params: 0

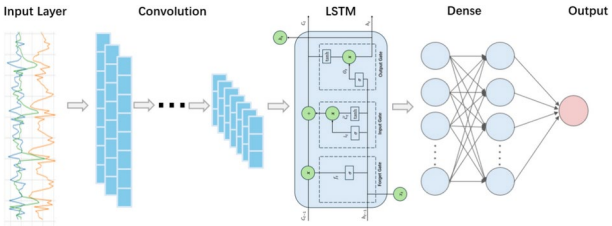


Fig. 12. Basic structure of DeepConv LSTM model.

2D CNN model

As shown in Fig. 11, the basic architecture of the 2D CNN model includes 2D convolutional layers, pooling layers, and fully connected layers. The pooling layers are used to reduce feature dimensions, improve computational efficiency, and lower the risk of overfitting. Since 2D convolution can effectively extract spatial features[45], it is applied in human activity recognition tasks to efficiently capture and identify complex temporal features.

In recent years, lightweight models such as MobileNet, ShuffleNet, and EfficientNet have been widely applied in computer vision and related fields. In[46], the authors evaluated MobileNet-V2 on the WISDM dataset, where MobileNet-V2 achieved an accuracy of 98.15%. However, its model size exceeded 6 MB, making it suitable only for high-performance embedded devices. Therefore, based on the model design in[47], we developed a 2D CNN model (Table 2) to enhance spatial feature extraction in human activity recognition (HAR). The input data is first reshaped from (100, 3) to (100, 3, 1) to enable 2D convolution operations, following the DeepSense framework[48]. The model consists of two sets of 2D convolutional layers, each containing two convolutional layers (32 and 64 filters, respectively) with a kernel size of (3,1) and "same" padding. A max-pooling layer (2,1) follows each set to reduce dimensionality while retaining key features. The extracted features are then flattened and passed through a fully connected layer (128 units, L2 regularization), followed by a dropout layer (rate = 0.5) to prevent overfitting[49]. The final classification is performed via a softmax layer. The model is trained using categorical cross-entropy loss and RMSprop optimizer, with a batch size of 64 for 50 epochs. ReduceLROnPlateau and EarlyStopping are employed to dynamically adjust learning rates and prevent overfitting.

DeepConv LSTM model

As shown in Fig. 12, the DeepConv LSTM model combines Conv1D layers with Long Short-Term Memory (LSTM) networks, enabling the simultaneous extraction of both spatial and temporal features. This dual feature

extraction capability gives the model an advantage in handling time-series data, compared to models that rely solely on spatial features, such as the 1D CNN and 2D CNN models.

The DeepConv LSTM model is one of the most popular models for time series analysis today. In the literature, Challa et al.[22] proposed a multi-branch CNN-BiLSTM model, which consists of three branches. Each branch contains two Conv1D layers with different kernel sizes (3, 7, and 11) to capture various local temporal dependencies. The output of each branch is processed through pooling and dropout layers before being connected to a BiLSTM layer, which captures long-term dependencies in the sequence. The model then outputs classification results through two fully connected layers. Khatun et al.[23] introduced a deep convolutional LSTM model with a self-attention mechanism for human activity recognition. This model includes a CNN, three LSTM layers, two attention layers, three batch normalization layers, five dropout layers, and two fully connected layers. García et al.[50] proposed a hybrid model combining depthwise separable convolutional neural networks (DS-CNN) with bidirectional LSTM (Bi-LSTM). However, bidirectional LSTM and self-attention mechanisms can increase model memory size and response latency. In[51], the authors designed the CNN-LSTM-DNN-TL model to address these issues. This model includes two Conv1D layers for feature extraction, followed by two LSTM layers to capture temporal dependencies, and finally outputs classification results through two dense layers. Based on these studies, a lightweight model suitable for resource-constrained environments was designed(Table 3).

The model begins with two Conv1D layers for initial feature extraction, with the first layer using 32 filters and the second layer using 64 filters, both with a kernel size of 3. Dropout layers with a rate of 0.3 are added after each convolutional layer to prevent overfitting. Conv1D layers are highly effective in extracting local spatial features, such as directional changes and intensity in accelerometer data, which are crucial for the initial classification of human activities. However, relying solely on convolutional layers may not sufficiently capture temporal dependencies in the data, as these layers mainly focus on short-term features. Therefore, a 100-unit LSTM layer is added to capture long-term dependencies in the data.

The model begins with two Conv1D layers (32 and 64 filters, kernel size = 3) to extract local spatial features from accelerometer data, followed by dropout layers (rate = 0.3) to prevent overfitting. To capture long-term dependencies, a 100-unit LSTM layer processes the sequential data. The extracted features are flattened and passed through a softmax layer for classification into six activity categories.The model is trained using categorical cross-entropy loss and the RMSprop optimizer (learning rate = 0.001), with a batch size of 64 for 50 epochs. ReduceLROnPlateau and EarlyStopping are applied to optimize training efficiency and prevent overfitting.

The advantage of LSTM is its ability to effectively capture long-term dependencies in the input data without significantly increasing computational complexity, which is crucial for accurately identifying and classifying complex human activities. For example, dynamic activities like walking and jogging exhibit significant temporal dependency features, and these features, over time, may contain important discriminative information. LSTM, through its memory cells and gating mechanisms, can learn these temporal dependency patterns, thereby improving the accuracy of the model.

LSTM overcomes the vanishing gradient problem commonly found in traditional RNN models, maintains long-term memory, and effectively handles long-term dependencies in data. The pioneering research by Hochreiter and Schmidhuber[42] demonstrated the superiority of LSTM in handling time-series data, especially in tasks requiring long-term memory. Additionally, the study by Sundermeyer et al.[52] showed that LSTM significantly outperforms traditional RNN and CNN models in tasks involving long-term dependencies, such as language modeling.

Performance evaluation

The performance of the model is mainly evaluated using accuracy, precision, recall, and F1 score. TP represents the number of true positives, FP represents the number of false positives, FN represents the number of false negatives, and TN represents the number of true negatives. The following equations describe how these metrics are calculated:

$$Precision = \frac{TP}{TP + FP} \tag{7}$$

$$Recall = \frac{TP}{TP + FN} \tag{8}$$

Layer	Output Shape	Param #
Conv1D	(None, 98, 32)	320
Dropout	(None, 98, 32)	0
Conv1D	(None, 96, 64)	6208
Dropout	(None, 96, 64)	0
LSTM	(None, 96, 100)	66000
Flatten	(None, 9600)	0
Dropout	(None, 9600)	0
Dense	(None, 6)	57606

Table 3. DeepConv LSTM Model summary. Total params: 130134 Trainable params: 130134 Non-trainable params: 0

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{9}$$

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{10}$$

Implement TinyML

This study primarily utilized TensorFlow (TF) and Keras. TensorFlow is a widely used open-source deep learning framework that supports deployment on various devices and platforms, including mobile, embedded, and edge devices. It also offers a rich set of tools and libraries for building and training neural networks[53]. By using TensorFlow Lite (TF-Lite), developers can run lightweight models on resource-constrained devices, enabling on-device machine learning[54]. Keras is a high-level neural network API known for its simple syntax and highly modular structure, specifically designed for building and training deep learning models[55]. As a high-level interface for TensorFlow, Keras is deeply integrated with TensorFlow, simplifying the processes of model definition, training, and evaluation, thus allowing researchers to focus more on model innovation and application development.

In this study, we employed full integer quantization[56], an aggressive quantization method that converts all values in the model, including both weights and activations, into integer representations.The quantization process follows the standard transformation:

$$q = \text{round}\left(\frac{f}{S}\right) + Z \tag{11}$$

where q is the quantized integer, f is the original floating-point value, S is the scale factor, and Z is the zero point.

To minimize any potential loss of accuracy, a representative calibration dataset is required to capture the activation range before quantization, ensuring the process is accurate. This approach significantly reduces the model's computational and memory requirements, making it highly suitable for deployment on low-power and memory-constrained microcontrollers and other embedded systems. Although full integer quantization may impact model accuracy, the trade-off is generally acceptable in extreme environments and resource-limited scenarios.

The quantized model was converted into a binary file using Edge Impulse, and the final binary file was flashed onto the Arduino Nano 33 BLE Sense Rev2 microcontroller. In the literature[25,33,57–59], the Arduino Nano 33 BLE Sense is commonly used due to its support for low-power Bluetooth connectivity and its built-in accelerometer, which facilitates the collection of triaxial accelerometer data. In this study, we chose the Arduino Nano 33 BLE Sense Rev2. There is no difference between the two versions in terms of microcontroller specifications. However, the Rev2 version offers several advantages: it is more cost-effective, has a more stable hardware design, and is better compatible with the latest Arduino IDE and libraries[60]. The steps from data processing to deployment are illustrated in Fig. 13.

Results

Model evaluation

Through experiments with various window sizes for the three models (as shown in Table 4), we found that a 5-second window provided optimal accuracy and feature extraction. Therefore, in this section, 100 raw samples are aggregated within each window to generate new features.The performance of the 1D CNN, DeepConv LSTM, and 2D CNN models is evaluated using accuracy, precision, recall, F1 score, and the confusion matrix.

Classification using 1D CNN

In terms of performance evaluation, the 1D CNN model achieved an accuracy of 96.88% and an F1 score of 0.9688, demonstrating high classification capability. The model has 378,054 trainable parameters, with a size

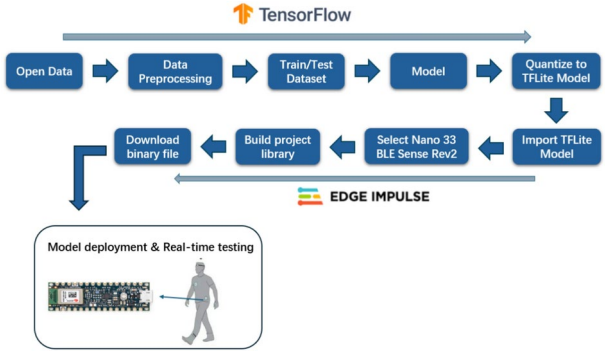


Fig. 13. From data preprocessing to model preparation, the trained model is then quantized to TensorFlow Lite. Finally, it is deployed onto the Arduino Nano 33 BLE Sense Rev2 using Edge Impulse.

Model	20(1s)	40(2s)	64(3.2s)	80(4s)	100(5s)	128(6.4s)	200(10s)
1D CNN	0.9353	0.9451	0.9624	0.9582	**0.9688**	0.9557	0.9421
2D CNN	0.9167	0.9352	0.9539	0.9587	**0.9666**	0.9633	0.9599
DeepConv LSTM	0.9390	0.9547	0.9668	0.9585	**0.9824**	0.9718	0.9675

Table 4. The accuracy of the three models under various window sizes.

Class	Precision	Recall	F1-score	Support
Downstairs	0.92	0.90	0.91	686
Jogging	0.99	0.98	0.98	2156
Sitting	0.98	0.99	0.98	377
Standing	0.99	0.99	0.99	323
Upstairs	0.89	0.90	0.90	811
Walking	0.99	0.99	0.99	2733
Accuracy			0.97	7086
Macro avg	0.96	0.96	0.96	7086
Weighted avg	0.97	0.97	0.97	7086

Table 5. 1D CNN Performance Metrics.

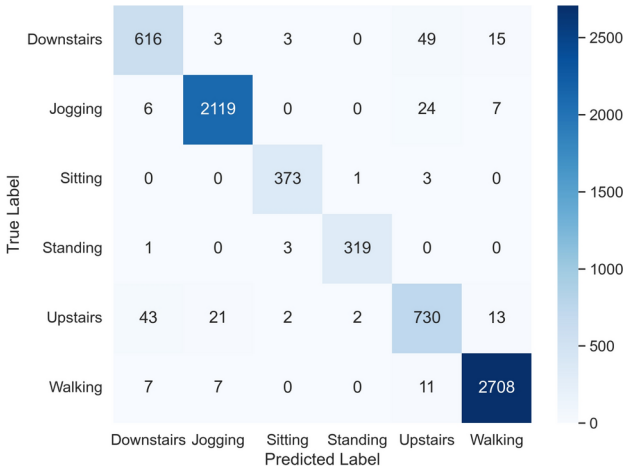


Fig. 14. Confusion matrix of 1D CNN model.

of 1483.02 KB, which was reduced to 381.73 KB after quantization to uint8 format. Each epoch of training took approximately 2 seconds. As shown in Table 5, the model exhibited precision and recall rates of 0.99 for recognizing activities such as 'Walking" and "Standing." However, the model experienced some misclassifications when distinguishing between the "Downstairs" and "Upstairs" categories, indicating that there is room for improvement in differentiating these two categories. This is further confirmed by the confusion matrix in Fig. 14, where the "Downstairs" category is frequently misclassified as "Upstairs." These results suggest that while the 1D CNN model performs well overall, there is still room for optimization in handling certain subtle activity categories.

Classification using 2D CNN

The 2D CNN model extracts features from data using two-dimensional convolutional and pooling layers, achieving an accuracy of 96.10%. The model has 637,094 trainable parameters, with a size of 2493.77 KB, which was reduced to 633.47 KB after quantization to uint8 format. Each epoch of training took approximately 2 seconds. As shown in Table 6, the model reached a precision and recall values of 0.99 for the "Standing" category. However, it only achieved a precision of 0.85 for the "Downstairs" category and a recall of 0.84 for the "Upstairs" category. The confusion matrix (Fig. 15) indicates that some samples from the "Downstairs" category were misclassified as "Upstairs," and vice versa. These misclassifications may stem from the high similarity in features between these two activities, resulting in the model's insufficient performance in distinguishing such subtle differences. This type of error suggests that the model still struggles to accurately identify certain activities, particularly when dealing with more challenging categories.

Class	Precision	Recall	F1-score	Support
Downstairs	0.85	0.90	0.88	686
Jogging	0.98	0.98	0.98	2156
Sitting	0.98	0.99	0.99	377
Standing	0.99	0.99	0.99	323
Upstairs	0.90	0.84	0.87	811
Walking	0.98	0.99	0.99	2733
Accuracy			0.96	7086
Macro avg	0.95	0.95	0.95	7086
Weighted avg	0.96	0.96	0.96	7086

Table 6. 2D CNN Performance Metrics.

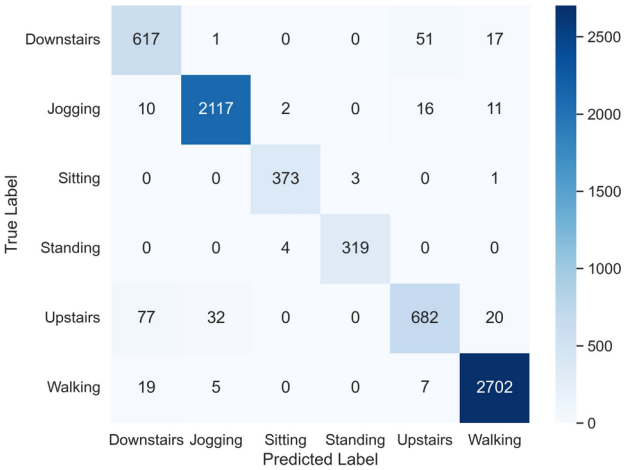


Fig. 15. Confusion matrix of 2D CNN model.

Classification using DeepConv LSTM

The DeepConv LSTM model combines Conv1D layers with LSTM networks to capture both spatial and temporal features, achieving an accuracy of 98.24% and an F1 score of 98.23%. The model has 130,134 trainable parameters, with a size of 513.23 KB, which was reduced to 136.51 KB after quantization to uint8 format. Each epoch of training took approximately 5 seconds, demonstrating the model's efficiency in terms of both storage and computational requirements. In terms of performance evaluation, the DeepConv LSTM model excels in the human activity recognition task by leveraging the strengths of both CNN and LSTM. The model utilizes two Conv1D layers to extract local features from the input data, followed by an LSTM layer to capture long-term dependencies in the time-series data. This architecture enables the model to focus on both spatial patterns and temporal dependencies, making it highly suitable for processing time-series data, such as human activity recognition.

Specifically, the confusion matrix (Fig. 16) shows that the DeepConv LSTM model demonstrates high precision and recall rates, indicating excellent differentiation capabilities for these categories. The evaluation metrics in Table 7 further support this, with the model achieving an accuracy of 98.24% and an F1 score of 0.9823 on the test dataset. These results indicate that the DeepConv LSTM model can not only accurately classify most activity categories but also effectively reduce the occurrence of misclassifications.

Although the 1D CNN and 2D CNN models perform well in human activity recognition tasks, they show certain limitations when handling subtle activity categories, particularly with noticeable misclassification between the "Downstairs" and "Upstairs" categories. These misclassifications are primarily due to the high similarity in movement patterns between these activities, which leads to overlapping feature representations in the learned embeddings.Additionally, these models tend to capture spatial patterns without fully leveraging the temporal dependencies in the data, further contributing to the challenge of distinguishing these two activities.

Table 8 compares the proposed model with existing lightweight models in terms of average accuracy, precision, recall, and F1 score. In contrast, the DeepConv LSTM model handles temporal dependencies in time-series data more effectively, outperforming existing models in both accuracy and model size. The introduction of LSTM layers allows the model to retain and utilize long-term dependencies, enhancing its ability to distinguish activities with subtle differences, such as "Downstairs" and "Upstairs". Moreover, the sequential nature of LSTM processing allows the model to better capture transition patterns in movement, which plays a crucial role in distinguishing these challenging activity classes.

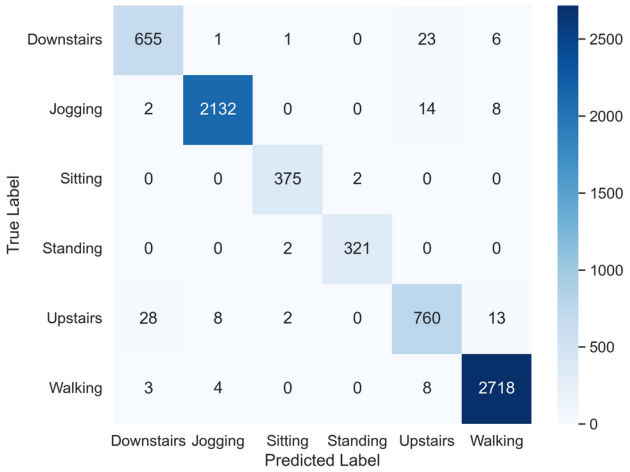


Fig. 16. Confusion matrix of DeepConv LSTM model.

Class	Precision	Recall	F1-score	Support
Downstairs	0.95	0.95	0.95	686
Jogging	0.99	0.99	0.99	2156
Sitting	0.99	0.99	0.99	377
Standing	0.99	0.99	0.99	323
Upstairs	0.94	0.94	0.94	811
Walking	0.99	0.99	0.99	2733
Accuracy			0.98	7086
Macro avg	0.98	0.98	0.98	7086
Weighted avg	0.98	0.98	0.98	7086

Table 7. DeepConv LSTM Performance Metrics.

Model	Avg. Accuracy	Avg. Precision	Avg. Recall	Avg. F1-score	Model size(kb)
CNN(2023)[24]	0.9574	0.945	0.9133	0.9283	164
CNNLSTM2D(2023)[24]	0.9653	0.9517	0.945	0.9483	218
CNN+LSTM(2023)[24]	0.9768	0.955	0.9633	0.965	294
MHCNLS(2024)[25]	0.9817	0.962	0.9591	0.9621	-
1D-CNN(2023)[26]	0.9775	–	–	0.978	-
CNBILSGR(2023)[61]	0.9712	–	–	–	-
DACDNET(2024)[62]	0.976	–	–	–	-
CNN-FCM-LSTM(2024)[63]	0.9727	–	–	–	1065
VGG16[64]	0.8932	–	–	–	154(MB)
MobileNet[64]	0.8882	–	–	–	23.96(MB)
MobileNetV2[64]	0.7471	–	–	–	26.91(MB)
MobileNetV3 Small[64]	0.8247	–	–	–	11.6(MB)
EfficientNet B0[64]	0.8911	–	–	–	45.7(MB)
ResNet 18[64]	0.8753	–	–	–	15.41(MB)
MarNASNet-B(2023)[64]	0.9062	–	–	–	430.08
1D CNN(ours)	0.9688	0.9688	0.9688	0.9688	381.73
2D CNN(ours)	0.961	0.9611	0.961	0.9609	633.47
DeepConv+LSTM(ours)	**0.9824**	**0.9823**	**0.9824**	**0.9823**	**136.51**

Table 8. Comparison of model performance.

In summary, the DeepConv LSTM model performs excellently in this study because it effectively combines spatial and temporal features to achieve accurate recognition of complex activities. Compared to the 1D and 2D CNN models, it shows significant advantages in accuracy and reducing misclassification rates, making it the best choice for human activity recognition tasks.

XAI Results

Explainable AI (XAI) refers to techniques that enhance the interpretability and transparency of machine learning models, providing insights into their decision-making processes. XAI plays a crucial role in human activity recognition (HAR) tasks, as it helps researchers and end-users understand how the model differentiates between activities and identify potential sources of misclassification.

To visualize the feature space learned by the DeepConv LSTM model, we adopted the t-SNE (t-Distributed Stochastic Neighbor Embedding) method proposed in[65]. t-SNE is a dimensionality reduction technique that maps high-dimensional features to a two-dimensional plane while preserving local data structure. In the resulting visualization, similar points in the high-dimensional space are mapped close together, while points from different classes remain well separated.

As shown in Fig. 17, the t-SNE visualization reveals several interesting patterns. Walking and Jogging form distinct clusters, indicating that the model effectively differentiates them based on their unique motion characteristics. However, Downstairs and Upstairs exhibit significant overlap, underscoring the challenge of differentiation, as they exhibit similar acceleration and gait characteristics. This overlap may explain some of the misclassifications observed in the confusion matrix (Fig. 16). Nonetheless, the model demonstrates robust feature extraction capabilities, as most activities form clear and well-separated clusters.

Deployment on the microcontroller

In this study, we chose to deploy the DeepConv LSTM model on resource-constrained embedded devices. To reduce the memory footprint and computational burden on microcontrollers, we applied full integer quantization to the model. As shown in Fig. 18a, the figure illustrates the memory footprint of the model parameters after quantization.

The quantization process significantly reduced the model size, compressing the DeepConv LSTM model to just one-quarter of its original size, resulting in a storage requirement of only 136.51 KB. This reduction not only lessens storage demands but also decreases reliance on floating-point computations, thereby improving inference efficiency. Although quantization caused a slight decrease in accuracy from 98.24% to 97.09% (as shown in Fig. 18b), this minor loss in accuracy is outweighed by the substantial reduction in model size and computational resource requirements.

The quantized model was imported into the Edge Impulse platform, as shown in (Fig. 19). The platform automatically calculates the required RAM, ROM, and latency for the model on different devices. After loading the model, the platform converts the TFLite model into a binary file, which is then deployed to the Arduino Nano 33 BLE Sense Rev2 device. During deployment, we conducted a comprehensive evaluation of its performance, including inference time, memory usage, and flash storage. The test results show that the average inference time of the DeepConv LSTM model is 21 ms. We evaluated the computational complexity of the proposed model following the method described in[66]. The total number of multiply-accumulate operations (MACs) is approximately 6.973×10^6, corresponding to about 1.395×10^7 operations. This results in a total of 0.01395 GOP. Dividing by the average inference time yields a computational performance of 0.664 GOPS. These results indicate that the model has low computational complexity and is well-suited for deployment on resource-constrained devices. On the Arduino Nano 33 BLE Sense Rev2, the model utilized 29.1 KB of memory and 189.6 KB of flash storage. Additionally, we measured power consumption using a USB voltage and current meter. Multiple tests were conducted and averaged to ensure measurement reliability. The results show that the device

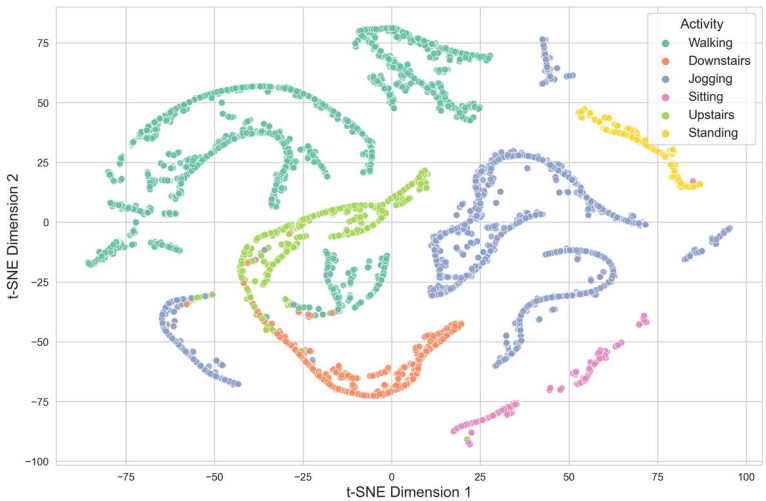
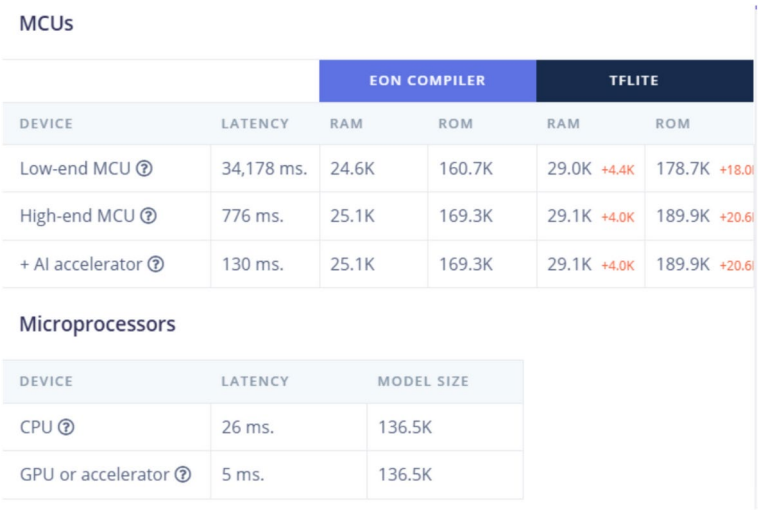


Fig. 17. t-SNE visualization of the feature space learned by the DeepConv LSTM model.

Fig. 18. (**a**) Size comparison of TensorFlow Models. (**b**) TinyML Confusion Matrix.



MCUs

		EON COMPILER		TFLITE	
DEVICE	LATENCY	RAM	ROM	RAM	ROM
Low-end MCU	34,178 ms.	24.6K	160.7K	29.0K +4.4K	178.7K +18.0
High-end MCU	776 ms.	25.1K	169.3K	29.1K +4.0K	189.9K +20.6
+ AI accelerator	130 ms.	25.1K	169.3K	29.1K +4.0K	189.9K +20.6

Microprocessors

DEVICE	LATENCY	MODEL SIZE
CPU	26 ms.	136.5K
GPU or accelerator	5 ms.	136.5K

Fig. 19. On-device performance. Low-end MCU: Estimate for a Cortex-M0+ or similar, running at 40MHz. High-end MCU: Estimate for a Cortex-M7 or other high-end MCU/DSP, running at 240MHz. +AI accelerator: Estimate for an MCU plus neural network accelerator.

operates at a voltage of 5.232V and a current of 0.004A, yielding an average power consumption of only 0.021W. Based on these results, we estimate that with a 1000mAh lithium battery, the device can operate continuously for up to 7 days under 24-hour usage. If the daily operation is limited to 8 hours, the battery life can extend to approximately 22 days. This demonstrates the model's energy efficiency and its suitability for long-term deployment on resource-constrained embedded platforms. While the inference time on this microcontroller is approximately 2 seconds, the model is still able to meet the requirements for real-time applications and maintain efficient operation.

Discussion

This study utilized a dataset provided by the WISDM Lab at the Department of Computer and Information Science, Fordham University, Bronx, New York. During the preprocessing phase, the raw data was aggregated into 100 samples per 5-second segment to generate new features, and a 50% overlapping sliding window was applied to retain information from the previous window. We designed and evaluated three different deep learning models to identify the most suitable model for deployment on edge devices. The experimental results indicate that the DeepConv LSTM model outperformed the 1D CNN and 2D CNN models, achieving a classification accuracy of 98.24% and a validation loss of 0.0699.

After full integer quantization, the size of the DeepConv LSTM model was reduced to 136.51 KB, with RAM usage of 29.1 KB, flash memory usage of 189.6 KB, and an average inference time of 21 milliseconds. The model's accuracy slightly decreased from 98.24% to 97.09%. Due to the resource constraints of the Arduino Nano 33 BLE Sense Rev2, the inference time for real-time recognition was approximately 2 seconds. Considering that our

window size is 5 seconds, this makes the on-board processing functionally feasible. This edge deployment aims to reduce reliance on cloud services and enhance user privacy. Additionally, the model's ability to process time-dependent data demonstrates its broad application potential, further validating the feasibility of using TinyML technology to handle complex computational tasks in resource-constrained environments.

Although the DeepConv LSTM model achieved excellent performance on the WISDM dataset, it has not been tested on other datasets or deployed on resource-constrained devices with varying capabilities. In future research, the model could be evaluated on multiple datasets to ensure its generalizability. Additionally, the WISDM dataset is highly imbalanced, which can affect the model's performance on minority classes. and techniques such as data augmentation or oversampling could be explored to improve the model's performance on minority classes.

Data availability

The WISDM Activity Prediction dataset (v1.1) used in this study is publicly available and can be accessed from the Wireless Sensor Data Mining (WISDM) Lab at Fordham University: http://www.cis.fordham.edu/wisdm/dataset.php.

Received: 11 September 2024; Accepted: 14 April 2025
Published online: 22 April 2025

References

1. Statista. Internet of things (iot) - statistics & facts. *figshare* https://www.statista.com/topics/2637/internet-of-things/#editorsPicks (2014).
2. Zhu, Y., Zhong, N. & Xiong, Y. Data explosion, data nature and dataology. In *Brain Informatics: International Conference, BI 2009 Beijing, China, October 22–24, 2009 Proceedings* (ed. Zhu, Y.) 147–158 (Springer, 2009).
3. Statista. Level 4/5 autonomous vehicles as a share of total vehicle sales in selected regions in 2035, by scenario. https://www.statista.com/statistics/1230752/level-4-5-autonomous-vehicles-share-total-vehicle-sales-selected-regions-scenario/. Last accessed 21-Aug-2024.
4. Shafiq, M., Tian, Z., Bashir, A. K., Du, X. & Guizani, M. Corrauc: A malicious bot-iot traffic detection method in iot network using machine-learning techniques. *IEEE Internet Things J.* **8**, 3242–3254. https://doi.org/10.1109/JIOT.2020.3002255 (2021).
5. T, S. *et al.* Analyzing the effect of edge computing on real-time data processing and latency reduction. In *2023 10th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON)*, vol. 10, 623–627, https://doi.org/10.1109/UPCON59197.2023.10434652 (2023).
6. Qiu, T. et al. Edge computing in industrial internet of things: Architecture, advances and challenges. *IEEE Commun. Surv. Tutor.* **22**, 2462–2488. https://doi.org/10.1109/COMST.2020.3009103 (2020).
7. Wang, J., Chen, Y., Hao, S., Peng, X. & Hu, L. Deep learning for sensor-based activity recognition: A survey. *Pattern Recogn. Lett.* **119**, 3–11. https://doi.org/10.1016/j.patrec.2018.02.010 (2019).
8. Wang, X. & Shang, J. Human activity recognition based on two-channel residual-gru-eca module with two types of sensors. *Electronics* https://doi.org/10.3390/electronics12071622 (2023).
9. Porzi, L., Messelodi, S., Modena, C. M. & Ricci, E. A smart watch-based gesture recognition system for assisting people with visual impairments. In *IMMPD '13* (2013).
10. Lara, O. D. & Labrador, M. A. A survey on human activity recognition using wearable sensors. *IEEE Commun. Surv. Tutor.* **15**, 1192–1209. https://doi.org/10.1109/SURV.2012.110112.00192 (2013).
11. Chaquet, J. M., Carmona, E. J. & Fernández-Caballero, A. A survey of video datasets for human action and activity recognition. *Comput. Vis. Image Underst.* **117**, 633–659. https://doi.org/10.1016/j.cviu.2013.01.013 (2013).
12. Chen, Z., Zhang, L., Jiang, C., Cao, Z. & Cui, W. Wifi csi based passive human activity recognition using attention based blstm. *IEEE Trans. Mob. Comput.* **18**, 2714–2724. https://doi.org/10.1109/TMC.2018.2878233 (2019).
13. Chen, K. et al. Deep learning for sensor-based human activity recognition: Overview, challenges, and opportunities. *ACM Comput. Surv.* https://doi.org/10.1145/3447744 (2021).
14. Mekruksavanich, S. Medical expert system based ontology for diabetes disease diagnosis. In *2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, 383–389, https://doi.org/10.1109/ICSESS.2016.7883091 (2016).
15. Mekruksavanich, S. & Jitpattanakul, A. Exercise activity recognition with surface electromyography sensor using machine learning approach. In *2020 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI DAMT & NCON)*, 75–78, https://doi.org/10.1109/ECTIDAMTNCON48261.2020.9090711 (2020).
16. Ang, K.-S. & Tham, C.-K. Smartphone-based vehicular and activity sensing. In *2012 18th IEEE International Conference on Networks (ICON)*, 1–6, https://doi.org/10.1109/ICON.2012.6506524 (2012).
17. Figo, D. et al. Preprocessing techniques for context recognition from accelerometer data. *Pers. Ubiquit. Comput.* **14**, 645–662. https://doi.org/10.1007/s00779-010-0293-9 (2010).
18. Kwapisz, J. R., Weiss, G. M. & Moore, S. A. Activity recognition using cell phone accelerometers. *SIGKDD Explor. Newsl.* **12**, 74–82. https://doi.org/10.1145/1964897.1964918 (2011).
19. Zara, N. J., Rifat, R. R., Tasnim, J., Baker, M. & Khan, R. Analysis of machine learning models for wearable devices centric human activity recognition. In *2024 3rd International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, 439–444, https://doi.org/10.1109/ICAAIC60222.2024.10575575 (2024).
20. Ullah, M., Ullah, H., Khan, S. D. & Cheikh, F. A. Stacked lstm network for human activity recognition using smartphone data. In *2019 8th European Workshop on Visual Information Processing (EUVIP)*, 175–180, https://doi.org/10.1109/EUVIP47703.2019.8946180 (2019).
21. Gupta, S. Deep learning based human activity recognition (har) using wearable sensor data. *Int. J. Inf. Manag. Data Insights* **1**, 100046. https://doi.org/10.1016/j.jjimei.2021.100046 (2021).
22. Challa, S., Kumar, A. & Semwal, V. A multibranch cnn-bilstm model for human activity recognition using wearable sensor data. *Vis. Comput.* **38**, 4095–4109. https://doi.org/10.1007/s00371-021-02283-3 (2022).
23. Khatun, M. A. et al. Deep cnn-lstm with self-attention model for human activity recognition using wearable sensor. *IEEE J. Transl. Eng. Health Med.* **10**, 1–16. https://doi.org/10.1109/JTEHM.2022.3177710 (2022).
24. Özge Bursa, S., Özlem Durmaz, İ & Alptekin, G. I. Building lightweight deep learning models with TensorFlow lite for human activity recognition on mobile devices. *Ann. Telecommun.* **78**, 687–702. https://doi.org/10.1007/s12243-023-00962-x (2023).
25. Gaud, N., Rathore, M. & Suman, U. Mhcnls-har: Multiheaded cnn-lstm-based human activity recognition leveraging a novel wearable edge device for elderly health care. *IEEE Sens. J.* **24**, 35394–35405. https://doi.org/10.1109/JSEN.2024.3450499 (2024).

26. Kaya, Y. & Topuz, E. K. Human activity recognition from multiple sensors data using deep CNNs. *Multim. Tools Appl.* **83**, 10815–10838. https://doi.org/10.1007/s11042-023-15830-y (2024).
27. Lim, W.-S., Seo, W., Kim, D.-W. & Lee, J. Efficient human activity recognition using lookup table-based neural architecture search for mobile devices. *IEEE Access* **11**, 71727–71738. https://doi.org/10.1109/ACCESS.2023.3294564 (2023).
28. Shi, W. & Dustdar, S. The promise of edge computing. *Computer* **49**, 78–81. https://doi.org/10.1109/MC.2016.145 (2016).
29. Rong, G. et al. An edge-cloud collaborative computing platform for building aiot applications efficiently. *J. Cloud Comput. Adv. Syst. Appl.* https://doi.org/10.1186/s13677-021-00250-w (2021).
30. Oufettoul, H., Chaibi, R. & Motahhir, S. Tinyml applications, research challenges, and future research directions. In *2024 21st Learning and Technology Conference (L &T)*, 86–91, https://doi.org/10.1109/LT60077.2024.10469633 (2024).
31. Ray, P. P. A review on tinyml: State-of-the-art and prospects. *J. King Saud Univ. Comput. Inf. Sci.* **34**, 1595–1623. https://doi.org/10.1016/j.jksuci.2021.11.019 (2022).
32. Hammad, S. S., Iskandaryan, D. & Trilles, S. An unsupervised tinyml approach applied to the detection of urban noise anomalies under the smart cities environment. *Internet of Things* **23**, 100848. https://doi.org/10.1016/j.iot.2023.100848 (2023).
33. Srinivasagan, R., Mohammed, M. & Alzahrani, A. Tinyml-sensor for shelf life estimation of fresh date fruits. *Sensors* https://doi.org/10.3390/s23167081 (2023).
34. Antonini, M., Pincheira, M., Vecchio, M. & Antonelli, F. An adaptable and unsupervised tinyml anomaly detection system for extreme industrial environments. *Sensors* https://doi.org/10.3390/s23042344 (2023).
35. Luo, F., Khan, S., Huang, Y. & Wu, K. Binarized neural network for edge intelligence of sensor-based human activity recognition. *IEEE Trans. Mob. Comput.* **22**, 1356–1368. https://doi.org/10.1109/TMC.2021.3109940 (2023).
36. Bao, W. et al. Edge computing-based joint client selection and networking scheme for federated learning in vehicular iot. *China Commun.* **18**, 39–52. https://doi.org/10.23919/JCC.2021.06.004 (2021).
37. Wisdm: Wireless sensor data mining. https://www.cis.fordham.edu/wisdm/dataset.php. Last accessed 18-Aug-2024.
38. Balaha, H. M. & Hassan, A.E.-S. Comprehensive machine and deep learning analysis of sensor-based human activity recognition. *Neural Comput. Appl.* **35**, 12793–12831. https://doi.org/10.1007/s00521-023-08374-7 (2023).
39. Straczkiewicz, M., James, P. & Onnela, J.-P. A systematic review of smartphone-based human activity recognition methods for health research. *NPJ Digit. Med.***4** (2021).
40. Wang, G. et al. Impact of sliding window length in indoor human motion modes and pose pattern recognition based on smartphone sensors. *Sensors* https://doi.org/10.3390/s18061965 (2018).
41. Bashir, S. A., Doolan, D. C. & Petrovski, A. V. The effect of window length on accuracy of smartphone-based activity recognition. (2016).
42. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Computat.* **9**, 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735 (1997).
43. Cho, H. & Yoon, S. M. Divide and conquer-based 1d cnn human activity recognition using test data sharpening. *Sensors* https://doi.org/10.3390/s18041055 (2018).
44. Ragab, M. G., Abdulkadir, S. J. & Aziz, N. Random search one dimensional cnn for human activity recognition. In *2020 International Conference on Computational Intelligence (ICCI)*, 86–91, https://doi.org/10.1109/ICCI51257.2020.9247810 (2020).
45. Lecun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **86**, 2278–2324. https://doi.org/10.1109/5.726791 (1998).
46. Yu, J. et al. Repmobile: A mobilenet-like network with structural reparameterization for sensor-based human activity recognition. *IEEE Sens. J.* **24**, 24224–24237. https://doi.org/10.1109/JSEN.2024.3412736 (2024).
47. Raj, R. & Kos, A. An improved human activity recognition technique based on convolutional neural network. *Sci. Rep.* **13**, 22581. https://doi.org/10.1038/s41598-023-49739-1 (2023).
48. Yao, S., Hu, S., Zhao, Y., Zhang, A. & Abdelzaher, T. Deepsense: A unified deep learning framework for time-series mobile sensing data processing. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, 351-360, https://doi.org/10.1145/3038912.3052577 (International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 2017).
49. Yang, J., Nguyen, M. N., San, P. P., Li, X. & Krishnaswamy, S. Deep convolutional neural networks on multichannel time series for human activity recognition. In *International Joint Conference on Artificial Intelligence* (2015).
50. Garcia-Gonzalez, D., Rivero, D., Fernandez-Blanco, E. & Luaces, M. R. Deep learning models for real-life human activity recognition from smartphone sensor data. *Internet of Things* **24**, 100925. https://doi.org/10.1016/j.iot.2023.100925 (2023).
51. Hayajneh, A. M., Hafeez, M., Zaidi, S. A. R. & McLernon, D. Tinyml empowered transfer learning on the edge. *IEEE Open J. Commun. Soc.* **5**, 1656–1672. https://doi.org/10.1109/OJCOMS.2024.3373177 (2024).
52. Sundermeyer, M., Schlüter, R. & Ney, H. Lstm neural networks for language modeling. In *Interspeech* (2012).
53. Rzym, G., Masny, A. & Chołda, P. Dynamic telemetry and deep neural networks for anomaly detection in 6g software-defined networks. *Electronics* https://doi.org/10.3390/electronics13020382 (2024).
54. TensorFlow. Tensorflow lite guide. https://www.tensorflow.org/lite/guide. Last accessed 26-Aug-2024.
55. Keras. Keras getting started. https://keras.io/getting_started/. Last accessed 31-Aug-2024.
56. TensorFlow. Tensorflow lite post-training quantization. https://www.tensorflow.org/lite/performance/post_training_quantization?hl=zh-cn. Last accessed 26-Aug-2024.
57. Brennan, D. & Galvin, P. Evaluation of a machine learning algorithm to classify ultrasonic transducer misalignment and deployment using tinyml. *Sensors* https://doi.org/10.3390/s24020560 (2024).
58. Atanane, O., Mourhir, A., Benamar, N. & Zennaro, M. Smart buildings: Water leakage detection using tinyml. *Sensors* https://doi.org/10.3390/s23229210 (2023).
59. Ren, H., Anicic, D. & Runkler, T. A. Tinyol: Tinyml with online-learning on microcontrollers. In *2021 International Joint Conference on Neural Networks (IJCNN)*, 1–8, https://doi.org/10.1109/IJCNN52387.2021.9533927 (2021).
60. Arduino. Arduino nano 33 ble sense rev2 suggested libraries. https://docs.arduino.cc/hardware/nano-33-ble-sense-rev2/#suggested-libraries. Last accessed 26-Aug-2024.
61. Sain, M. K., Singha, J., Saini, S. & Semwal, V. B. Human action recognition using convbilstm-gru in indoor environment. In *2023 IEEE Global Conference on Artificial Intelligence and Internet of Things (GCAIoT)*, 179–186, https://doi.org/10.1109/GCAIoT61060.2023.10385107 (2023).
62. Ye, N., Zhang, L., Xiong, D., Wu, H. & Song, A. Accelerating activity inference on edge devices through spatial redundancy in coarse-grained dynamic networks. *IEEE Internet Things J.* **11**, 41273–41285. https://doi.org/10.1109/JIOT.2024.3458441 (2024).
63. Bodhe, R., Sivakumar, S., Sakarkar, G., Juwono, F. H. & Apriono, C. Outdoor activity classification using smartphone based inertial sensor measurements. *Multim. Tools Appl.* **83**, 76963–76989. https://doi.org/10.1007/s11042-024-18599-w (2024).
64. Kobayashi, S., Hasegawa, T., Miyoshi, T. & Koshino, M. Marnasnets: Toward cnn model architectures specific to sensor-based human activity recognition. *IEEE Sens. J.* **23**, 18708–18717. https://doi.org/10.1109/JSEN.2023.3292380 (2023).
65. Aquino, G., Costa, M. G. F. & Filho, C. F. F. C. Explaining and visualizing embeddings of one-dimensional convolutional models in human activity recognition tasks. *Sensors* https://doi.org/10.3390/s23094409 (2023).
66. Xia, M. et al. Sparknoc: An energy-efficiency fpga-based accelerator using optimized lightweight cnn for edge computing. *J. Syst. Archit.* **115**, 101991. https://doi.org/10.1016/j.sysarc.2021.101991 (2021).

Author contributions

Conceptualization, HT.Z. and XJ.Z.; methodology, HT.Z. and XJ.Z.; software, HT.Z.and Y.F.; validation, HT.Z. and XJ.Z.; formal analysis, HT.Z.and TD.Z.; investigation, HT.Z.; resources, HT.Z.; data curation, HT.Z.; writing---original draft preparation, HT.Z.; writing---review and editing, XJ.Z.; visualization, HT.Z.; supervision, LJ.X.; project administration, LJ.X. All authors have read and agreed to the published version of the manuscript.

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to L.X.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.