Wearable-based Human Activity Recognition with Spatio-Temporal Spiking Neural Networks

Yuhang Li, Ruokai Yin, Hyoungseob Park, Youngeun Kim, Priyadarshini Panda
Department of Electrical Engineering, Yale University
New Haven, CT 06511, USA
{yuhang.li, ruokai.yin, hyoungseob.park, youngeun.kim, priya.panda}@yale.edu

Abstract

We study the Human Activity Recognition (HAR) task, which predicts user daily activity based on time series data from wearable sensors. Recently, researchers use end-to-end Artificial Neural Networks (ANNs) to extract the features and perform classification in HAR. However, ANNs pose a huge computation burden on wearable devices and lack temporal feature extraction. In this work, we leverage Spiking Neural Networks (SNNs)—an architecture inspired by biological neurons—to HAR tasks. SNNs allow spatio-temporal extraction of features and enjoy low-power computation with binary spikes. We conduct extensive experiments on three HAR datasets with SNNs, demonstrating that SNNs are on par with ANNs in terms of accuracy while reducing up to 94% energy consumption. The code is publicly available in https://github.com/Intelligent-Computing-Lab-Yale/SNN_HAR

1 Introduction

With the rapid development of smart devices such as phones and fitness trackers, sensing user activities or behavioral insights becomes more important for healthcare purposes. In this case, Human Activity Recognition (HAR) [1, 2, 3] seeks to predict the user activities using the smart devices' sensors such as accelerometer, gyroscope, electroencephalogram (EEG) sensor, etc. The objective of HAR includes sports injury detection, well-being management, medical diagnosis, smart building solutions [4] and elderly care [5].

Traditionally, researchers use hand-crafted features and simple classifiers for HAR tasks. Yet this type of method requires expert knowledge to get high-quality features. More recently, deep learning has been introduced to use end-to-end feature extraction, as well as classification [6]. They use convolutional layers in the Artificial Neural Networks (ANNs) [7, 8, 9] and optimize the model with gradient descent. However, ANNs use full precision (*i.e.* 32-bit floating-point operations) computation and incur low sparsity, bringing huge computation complexity and energy consumption to wearable devices. In addition, ANNs use ReLU neurons that do not consider correlation in time. This choice may be sub-optimal, especially for time series data since it simply adapts the ANN regime from the image domain.

To overcome the above limitations, we utilize Spiking Neural Networks (SNNs) [10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20] combined with convolutional layers for dealing with time series data in HAR. The HAR can be benefited from SNNs in two aspects: (1) SNNs take advantage of binary spikes (either 0 or 1) and thus enjoy multiplication-free and highly sparse computation that lowers energy consumption on time-series data; (2) SNNs can inherently model the temporal dynamics in time series data. The spiking neurons from SNNs maintain a variable called the membrane potential through time. As long as the membrane potential exceeds a pre-defined threshold, the neuron will fire

Workshop on Learning from Time Series for Health, 36th Conference on Neural Information Processing Systems (NeurIPS 2022).

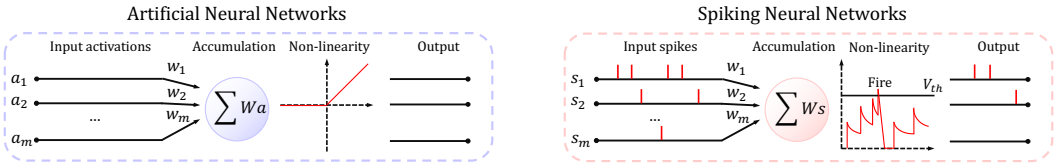


Figure 1: The schematic view of artificial neurons and spiking neurons. Artificial neuron takes full precision input and rectifies it if it is less than 0 and passes it otherwise; spiking neuron considers the correlation between times, and fires a spike only if the membrane potential is higher than a threshold.

a spike in the current time step. We verify our SNNs on three popular HAR datasets (UCI-HAR [3], UniMB SHAR [21], HHAR [22]) and compare them with ANNs baselines. Our SNNs can deliver the same or even higher accuracy than ANNs while reducing up to 94% energy consumption.

The rest of this paper is organized as follows: Sec. 2 describes the problem statement and our approach to spatio-temporal SNN. Sec. 3 provides the experimental results and visualization on three HAR datasets. Finally, we give our conclusion remark in Sec. 4.

2 Method

2.1 Notation and Problem Statement

In this paper, vectors/matrices are denoted with bold italic/capital letters (*e.g.* $\boldsymbol{x}$ and $\mathbf{W}$ represents the input vector and weight matrix). Constants are denoted by small upright letters.

Concretely, we denote the wearable-based sensor dataset with $\{\mathbf{x}_i\}_{i=1}^{N}$, and each sample $\mathbf{x}_i \in \mathbb{R}^{T\times D}$ is collected when the wearer is doing certain activity $\mathbf{y}_i$, *e.g.* running, sitting, lying, standing, *etc*. Here, data samples are streaming and have T time steps in total. D is the dimension of the sensor's output. As an example, the accelerometer records the acceleration in the (x, y, z)-axis, thus $D = 3$ for the accelerometer data. We are interested in designing an end-to-end model $f(\cdot)$ and optimizing it to predict the accurate activity label $\mathbf{y}$.

2.2 Spiking Neurons

We adopt the well-known Leaky-Integrate-and-Fire (LIF) neurons model for spiking neurons [23], which constantly receive inputs and outputs spikes through time. Formally, the LIF neuron maintains the membrane potential $\boldsymbol{v}$ through time, and suppose at t-th time step ($1 \le t \le T$), the membrane potential receives the pre-synaptic input charge $\boldsymbol{c}^{(t)}$, given by

$$\boldsymbol{v}^{(t+1),\text{pre}} = \tau \boldsymbol{v}^{(t)} + \boldsymbol{c}^{(t)}, \text{where } \boldsymbol{c}^{(t)} = \mathbf{W}\boldsymbol{s}^{(t)}. \tag{1}$$

Here, τ is a constant between $[0, 1]$ representing the decay factor of the membrane potential as time flows, which controls the correlation between time steps. $\tau = 0$ stands for 0 correlation and LIF degenerates to binary activation [24] without temporal dynamics, while $\tau = 1$ stands for maximum correlation and [25, 26] proves that LIF will become ReLU neuron when T is sufficiently large. $\boldsymbol{c}^{(t+1)}$ is the product between weights $\mathbf{W}$ and the spike $\boldsymbol{s}^{(t+1)}$ from last layer. After receiving the input charge, the LIF neuron will fire a spike if the pre-synaptic membrane potential exceeds some threshold, given by

$$\boldsymbol{s}^{(t+1)} = \begin{cases} 1 & \text{if } \boldsymbol{v}^{(t+1),\text{pre}} > V_{th} \\ 0 & \text{otherwise} \end{cases}, \tag{2}$$

where V_{th} is the firing threshold. Note that the spike $\boldsymbol{s}^{(t+1)}$ will propagate to the next layer, here we omit the layer index for simplicity.

If the LIF neurons fire the spike, the membrane potential will be reset. This can be done by either soft-reset or hard-reset, denoted by

$$\begin{cases} \boldsymbol{v}^{(t+1)} = \boldsymbol{v}^{(t+1),\text{pre}} \cdot (1 - \boldsymbol{s}^{(t+1)}) & \texttt{\# Hard-Reset} \\ \boldsymbol{v}^{(t+1)} = \boldsymbol{v}^{(t+1),\text{pre}} - \boldsymbol{s}^{(t+1)} \cdot V_{th} & \texttt{\# Soft-Reset} \end{cases}, \tag{3}$$

where hard-reset sets $\boldsymbol{v}^{(t+1)}$ to 0, while soft-reset subtracts $\boldsymbol{v}^{(t+1)}$ by V_{th}. We choose LIF neurons because the $s^{(t+1)}$ is binary and dependent on input in previous time steps. In our experiments, we will conduct ablation studies on the decay factor, the firing threshold, and the reset mechanism. In Fig. 1, we provide an overview difference between ANN and SNN neurons.

2.3 Integrating Spiking Neurons into Network

We integrate spiking neurons into deep neural networks by replacing their non-linear activation with LIF. Specifically, since the time series data naturally has a time dimension, we also integrate the pre-synaptic potential charge along this time dimension. For instance, suppose $\boldsymbol{a} \in \mathbb{R}^{n \times c \times T}$ is a pre-activation tensor, where n, c, T represent the batch size, channel number, and total time steps, respectively, we set the charge in each time step for LIF as the pre-activation in corresponding time step, *i.e.* $\boldsymbol{c}^{(t)} = \boldsymbol{a}_{:,:,t}$. Then, we stack the output spikes along the time dimension again, *i.e.* $\mathbf{S} = \mathrm{stack}(\{\boldsymbol{s}^{(t)}\}_{t=1}^{T})$, for calculating the pre-activation in next layer.

2.4 Optimization

Although LIF neurons manage to model the temporal features and produce binary spikes, the firing function (Eq. (2)) is discrete and thus produces zero gradients almost everywhere, prohibiting gradient-based optimization. Particularly, the gradient of loss (denoted by L) w.r.t. weights can be computed using the chain rule:

$$\frac{\partial L}{\partial \mathbf{W}} = \sum_{t=1}^{T} \frac{\partial L}{\partial \boldsymbol{s}^{(t)}} \frac{\partial \boldsymbol{s}^{(t)}}{\partial \boldsymbol{v}^{(t),\text{pre}}} \left(\frac{\partial \boldsymbol{v}^{(t),\text{pre}}}{\partial \boldsymbol{c}^{(t)}} \frac{\partial \boldsymbol{c}^{(t)}}{\partial \mathbf{W}} + \sum_{t'=1}^{t-1} \frac{\partial \boldsymbol{v}^{(t),\text{pre}}}{\partial \boldsymbol{v}^{(t)}} \frac{\partial \boldsymbol{v}^{(t)}}{\partial \boldsymbol{v}^{(t'),\text{pre}}} \frac{\partial \boldsymbol{v}^{(t'),\text{pre}}}{\partial \boldsymbol{c}^{(t')}} \frac{\partial \boldsymbol{c}^{(t')}}{\partial \mathbf{W}} \right). \tag{4}$$

Here, all other terms can be differentiated except $\frac{\partial \boldsymbol{s}^{(t)}}{\partial \boldsymbol{v}^{(t),\text{pre}}}$ which brings zero-but-all gradients. To circumvent this problem, we use the surrogate gradient method [27]. In detail, we use the triangle surrogate gradient, given by

$$\frac{\partial \boldsymbol{s}^{(t)}}{\partial \boldsymbol{v}^{(t),\text{pre}}} = \max\left(0, 1 - \left| \frac{\boldsymbol{v}^{(t),\text{pre}}}{V_{th}} - 1 \right| \right). \tag{5}$$

As a result, the SNNs can be optimized with stochastic gradient descent algorithms.

3 Experiments

In this section, we verify the effectiveness and efficiency of our SNNs on three popular HAR benchmarks. We first briefly provide the implementation details of our experiments and then compare our method with ANNs' baselines. Finally, we conduct ablation studies to validate our design choices.

3.1 Implementation Details

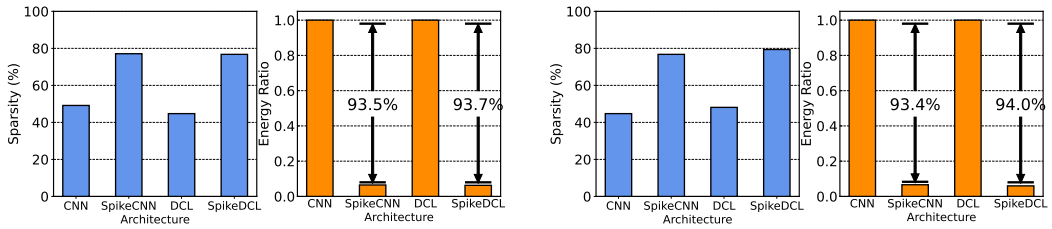
We implement our SNNs and existing ANNs with the PyTorch framework [28]. For all our experiments, we use Adam optimizer [29]. All models are trained for 60 epochs, with batch size 128. The only flexible hyper-parameter is the learning rate, which is selected from $\{1e-4, 3e-4, 1e-3\}$ with the best validation accuracy. We use Cosine Annealing Decay for the learning rate schedule. For all three HAR datasets, we split them to 64% as the training set, 16% as the validation set, and 20% as the test set. We report test accuracy when the model reaches the best validation accuracy. Note that these datasets only have one label for each input sample, therefore top-1 accuracy is the same as the F-1 score. The dataset descriptions are shown below:

UCI-HAR [3] contains 10.3k instances collected from 30 subjects. It involves 6 different activities including walking, walking upstairs, walking downstairs, sitting, standing, and lying. The sensors are the 3-axis accelerometer and 3-axis gyroscope (both are 50Hz) from Samsung Galaxy SII.

UniMB SHAR [21] contains 11.7k instances collected from 30 subjects. It involves 17 different activities including 9 kinds of daily living activities and 6 kinds of fall activities. The sensor is the 3-axis accelerometer (maximum 50Hz) from Samsung Galaxy Nexus I9250.

Table 1: Accuracy comparison between different networks on three HAR datasets (DCL=DeepConvLSTM).

Model	CNN	DCL	LSTM	Transformer	SpikeCNN	SpikeDCL
UCI-HAR [3]	96.29±0.12	97.87±0.32	82.41±4.04	96.02±0.27	**96.40±0.15**	**98.86±0.28**
SHAR [21]	92.38±0.51	90.78±1.05	83.87±0.96	83.19±0.74	**94.04±0.34**	**92.08±0.77**
HHAR [22]	96.19±0.14	97.15±0.17	95.59±0.20	95.82±0.16	**96.20±0.09**	**97.52±0.10**



(a) UCI-HAR **(b)** SHAR

Figure 2: Hardware performance comparison between ANN and SNN. The ANN's energy consumption is normalized to 1.

HHAR [22] contains 57k instances collected from 9 subjects. It involves 6 daily activities including biking, sitting, standing, walking, stair up, and stair down. The sensors are accelerometers from 8 smartphones and 4 smart watches (sampling rate from 50Hz to 200Hz).

3.2 Comparison with ANNs

Task Performance. For ANN baselines, we select CNN [30], DeepConvLSTM [31], LSTM [32], and Transformer [33] architectures. For our SNNs, we integrate them into CNN and DeepConvLSTM. The architecture specifications can be found in our code. Each result is averaged from 5 runs (random seeds from 1000 to 1004) and includes a standard deviation value. We summarize the results in Table 1, from which we find the SNNs have higher accuracy than the ANNs. For example, on the UniMB SHAR dataset, SpikeCNN has a 1.7% average accuracy improvement over its artificial CNN counterpart. Even more remarkably, the SpikeDeepConvLSTM (SpikeDCL) on the UCI-HAR dataset reaches 98.86% accuracy, which is 1% higher than DCL. Considering the accuracy is approaching 100%, the 1% improvement would be very significant. For UCI-HAR and HHAR datasets, we find SpikeCNN has similar accuracy to CNN, instead, the SpikeDeepConvLSTM consistently outperforms DeepConvLSTM, indicating that SNNs can be more coherent with the LSTM layer. Regarding the standard deviation of accuracy, we find that SNNs are usually more stable than ANNs, except for only one case, SpikeCNN on UCI-HAR.

Hardware Performance. Here, we compare two metrics, namely the activation sparsity and the energy consumption. Higher sparsity can avoid more computations with weights in hardware that supports sparse computation. We measure the sparsity either in ReLU (ANNs) or in LIF (SNNs) and visualize them in Fig. 2 left side. The ReLU in ANN usually has around 50% sparsity, an intuitive result since the mean of activation is usually around 0. LIF neurons, however, exhibit a higher sparsity, approximately 80%, probably due to the threshold for firing being larger than 0. As a result, the SNN has a higher potential to save more operations in inference.

The second metric in hardware performance is energy consumption. We estimate the energy consumption by evaluating the proposed SNN model together with our ReLU-based ANN baseline through the energy simulator proposed in [34]. Particularly, we estimate the energy reduction ratio on the hardware accelerator [34]. The results are shown in Fig. 2 right side. It can be seen that SNNs consume up to 94% less energy than ANNs, which could largely promote the battery life in smart devices. In summary, SNNs bring higher task performance due to the LIF neurons, and also energy efficiency due to the binary representation with high sparsity.

3.3 Ablation Studies

In this section, we conduct ablation studies with respect to the (hyper)-parameters in the LIF neurons, including decay factor, threshold, and reset mechanism. We test SpikeDCL and SpikeCNN on UCI-HAR and SHAR datasets.

Table 2: Ablation study on the decay factor τ.

Dataset	Model	Decay Factor τ				
		0.0	0.25	0.5	0.75	1.0
UCI-HAR [3]	SpikeCNN	95.48	95.63	95.78	**96.40**	95.92
	SpikeDCL	94.36	96.50	97.57	**98.86**	96.60
SHAR [21]	SpikeCNN	93.54	**94.04**	93.48	93.85	74.68
	SpikeDCL	89.53	**92.08**	90.93	90.10	60.55

Table 3: Ablation study on the firing threshold V_{th} and the reset mechanism.

Dataset	Model	Firing Threshold V_{th}				Reset	
		0.25	0.5	0.75	1.0	Hard	Soft
UCI-HAR [3]	SpikeCNN	95.71	**96.40**	96.18	96.11	96.09	**96.40**
	SpikeDCL	98.27	**98.86**	97.60	96.81	98.53	**98.73**
SHAR [21]	SpikeCNN	93.91	**94.04**	93.89	93.87	92.75	**94.04**
	SpikeDCL	91.42	**92.08**	91.72	91.53	91.13	**92.08**

Decay Factor. We select 5 fixed decay factors from $\{0.0, 0.25, 0.5, 0.75, 1.0\}$. Note that as discussed before $\tau = 0$ indicates no correlation between two consecutive time steps, therefore SNN becomes equivalent to Binary Activation Networks (BAN), while $\tau = 1$ indicates full correlation. We provide all results in Table 2. We can find that τ has a huge impact on the final test accuracy. For the UCI-HAR dataset with SpikeDCL, the accuracy of $\tau = 0$ is 94.36% while the accuracy of $\tau = 0.75$ is 98.86%. Additionally, if we compare other $0 < \tau < 1$ cases with $\tau = 0$, we find that $\tau = 0$ always produces a large deficiency. *This indicates that considering the temporal correlation with $\tau > 0$ is necessary for the time series tasks.* Moreover, for the SHAR dataset, the $\tau = 1$ only has 60.55% accuracy while the $\tau = 0.25$ case achieves 91.72% accuracy.

Firing Threshold. We next study the effect of the firing threshold. Generally, the firing threshold is related to the easiness of firing a spike. We set the threshold in $\{0.25, 0.5, 0.75, 1.0\}$ and run the same experiments with the former ablation. Here, through Table 3 we observe that the firing threshold has a unified pattern. SNN reaches its highest performance when the firing threshold is set to 0.5. This result is not surprising since 0.5 is in the mid of 0 and 1, and thus has the lowest error for the firing function (see Eq. (2)). Meanwhile, we find the difference in accuracy brought by the firing threshold is lower than the decay factor. For instance, the largest gap when changing the threshold for SpikeDCL on the SHAR dataset is 0.65%, while this gap can be 32% when changing the decay factor. Therefore, the SNN is more sensitive to the decay factor rather than the threshold.

Reset Mechanism. Finally, we verify the reset mechanism for SNNs, namely soft-reset and hard-reset. The results are sorted in the Table 3 as well. For all cases, the soft-reset mechanism is better than the hard-reset. We think the reason behind this is that the hard reset will directly set the membrane potential to 0, therefore cutting off the correlation between two time steps. Instead, the soft reset keeps some information on membrane potential after firing.

4 Conclusion

In this paper, we have introduced Spiking Neural Networks (SNNs) for HAR tasks, which, to our best knowledge, is the first of its kind study. Compared to the original Artificial Neural Networks (ANNs), SNNs utilize their LIF neurons to generate spikes through time, bringing energy efficiency as well as temporally correlated non-linearity. Our results show that SNNs achieve competitive accuracy while reducing energy significantly.

References

[1] Oscar D Lara and Miguel A Labrador. A survey on human activity recognition using wearable sensors. *IEEE communications surveys & tutorials*, 15(3):1192–1209, 2012.

[2] Michalis Vrigkas, Christophoros Nikou, and Ioannis A Kakadiaris. A review of human activity recognition methods. *Frontiers in Robotics and AI*, 2:28, 2015.

[3] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra Perez, and Jorge Luis Reyes Ortiz. A public domain dataset for human activity recognition using smartphones. In *Proceedings of the 21th international European symposium on artificial neural networks, computational intelligence and machine learning*, pages 437–442, 2013.

[4] E Ramanujam, Thinagaran Perumal, and S Padmavathi. Human activity recognition with smartphone and wearable sensors using deep learning techniques: A review. *IEEE Sensors Journal*, 21(12):13029–13040, 2021.

[5] Henry Friday Nweke, Ying Wah Teh, Ghulam Mujtaba, and Mohammed Ali Al-Garadi. Data fusion and multiple classifier systems for human activity detection and health monitoring: Review and open research directions. *Information Fusion*, 46:147–170, 2019.

[6] Henry Friday Nweke, Ying Wah Teh, Mohammed Ali Al-Garadi, and Uzoma Rita Alo. Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges. *Expert Systems with Applications*, 105:233–261, 2018.

[7] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

[8] Andrey Ignatov. Real-time human activity recognition from accelerometer data using convolutional neural networks. *Applied Soft Computing*, 62:915–922, 2018.

[9] Shaohua Wan, Lianyong Qi, Xiaolong Xu, Chao Tong, and Zonghua Gu. Deep learning models for real-time human activity recognition with smartphones. *Mobile Networks and Applications*, 25(2):743–755, 2020.

[10] Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothée Masquelier, and Anthony Maida. Deep learning in spiking neural networks. *Neural networks*, 111:47–63, 2019.

[11] Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, 2019.

[12] Lei Deng, Yujie Wu, Xing Hu, Ling Liang, Yufei Ding, Guoqi Li, Guangshe Zhao, Peng Li, and Yuan Xie. Rethinking the performance comparison between snns and anns. *Neural Networks*, 121:294 – 307, 2020.

[13] Priyadarshini Panda, Sai Aparna Aketi, and Kaushik Roy. Toward scalable, efficient, and accurate deep spiking neural networks with backward residual connections, stochastic softmax, and hybridization. *Frontiers in Neuroscience*, 14:653, 2020.

[14] Dennis Valbjørn Christensen, Regina Dittmann, Bernabé Linares-Barranco, Abu Sebastian, Manuel Le Gallo, Andrea Redaelli, Stefan Slesazeck, Thomas Mikolajick, Sabina Spiga, Stephan Menzel, et al. 2022 roadmap on neuromorphic computing and engineering. *Neuromorphic Computing and Engineering*, 2022.

[15] Youngeun Kim, Yuhang Li, Hyoungseob Park, Yeshwanth Venkatesha, and Priyadarshini Panda. Neural architecture search for spiking neural networks. *arXiv preprint arXiv:2201.10355*, 2022.

[16] Youngeun Kim, Yuhang Li, Hyoungseob Park, Yeshwanth Venkatesha, Ruokai Yin, and Priyadarshini Panda. Exploring lottery ticket hypothesis in spiking neural networks. In *European Conference on Computer Vision*, pages 102–120. Springer, 2022.

[17] Youngeun Kim and Priyadarshini Panda. Revisiting batch normalization for training low-latency deep spiking neural networks from scratch. *Frontiers in neuroscience*, page 1638.

[18] Youngeun Kim, Yeshwanth Venkatesha, and Priyadarshini Panda. Privatesnn: privacy-preserving spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 1192–1200, 2022.

[19] Shikuang Deng, Yuhang Li, Shanghang Zhang, and Shi Gu. Temporal efficient training of spiking neural network via gradient re-weighting. *arXiv preprint arXiv:2202.11946*, 2022.

[20] Yuhang Li, Shikuang Deng, Xin Dong, and Shi Gu. Converting artificial neural networks to spiking neural networks via parameter calibration. *arXiv preprint arXiv:2205.10121*, 2022.

[21] Daniela Micucci, Marco Mobilio, and Paolo Napoletano. Unimib shar: A dataset for human activity recognition using acceleration data from smartphones. *Applied Sciences*, 7(10):1101, 2017.

[22] Allan Stisen, Henrik Blunck, Sourav Bhattacharya, Thor Siiger Prentow, Mikkel Baun Kjærgaard, Anind Dey, Tobias Sonne, and Mads Møller Jensen. Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition. In *Proceedings of the 13th ACM conference on embedded networked sensor systems*, pages 127–140, 2015.

[23] Ying-Hui Liu and Xiao-Jing Wang. Spike-frequency adaptation of a generalized leaky integrate-and-fire model neuron. *Journal of computational neuroscience*, 10(1):25–45, 2001.

[24] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016.

[25] Yuhang Li, Shikuang Deng, Xin Dong, Ruihao Gong, and Shi Gu. A free lunch from ann: Towards efficient, accurate spiking neural networks calibration. In *International Conference on Machine Learning*, pages 6316–6325. PMLR, 2021.

[26] Shikuang Deng and Shi Gu. Optimal conversion of conventional artificial neural networks to spiking neural networks. *arXiv preprint arXiv:2103.00476*, 2021.

[27] Yuhang Li, Yufei Guo, Shanghang Zhang, Shikuang Deng, Yongqing Hai, and Shi Gu. Differentiable spike: Rethinking gradient-descent for training spiking neural networks. *Advances in Neural Information Processing Systems*, 34:23426–23439, 2021.

[28] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[29] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[30] Carlos Avilés-Cruz, Andrés Ferreyra-Ramírez, Arturo Zúñiga-López, and Juan Villegas-Cortéz. Coarse-fine convolutional deep-learning strategy for human activity recognition. *Sensors*, 19(7):1556, 2019.

[31] Debadyuti Mukherjee, Riktim Mondal, Pawan Kumar Singh, Ram Sarkar, and Debotosh Bhattacharjee. Ensemconvnet: a deep learning approach for human activity recognition using smartphone sensors for healthcare applications. *Multimedia Tools and Applications*, 79(41):31663–31690, 2020.

[32] LuKun Wang and RuYue Liu. Human activity recognition based on wearable sensor using hierarchical deep lstm networks. *Circuits, Systems, and Signal Processing*, 39(2):837–856, 2020.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[34] Ruokai Yin, Abhishek Moitra, Abhiroop Bhattacharjee, Youngeun Kim, and Priyadarshini Panda. Sata: Sparsity-aware training accelerator for spiking neural networks. *arXiv preprint arXiv:2204.05422*, 2022.