Efficient Human Activity Recognition Using Hyperdimensional Computing

Yeseong Kim
University of California
San Diego
yek048@ucsd.edu

Mohsen Imani
University of California
San Diego
moimani@ucsd.edu

Tajana S. Rosing
University of California
San Diego
tajana@ucsd.edu

ABSTRACT

Human activity recognition is a key task of many Internet of Things (IoT) applications to understand underlying contexts and react with the environments. Machine learning is widely exploited to identify the activities from sensor measurements, however, they are often overcomplex to run on less-powerful IoT devices. In this paper, we present an alternative approach to efficiently support the activity recognition tasks using brain-inspired hyperdimensional (HD) computing. We show how the HD computing method can be applied to the recognition problem in IoT systems while improving the accuracy and efficiency. In our evaluation conducted for three practical datasets, the proposed design achieves the speedup of the model training by up to 486x as compared to the state-of-the-art neural network training. In addition, our design improves the performance of the HD-based inference procedure by 7x on a low-power ARM processor.

Author Keywords

Human activity recognition, hyperdimensional computing, alternative computing

INTRODUCTION

Human-aware system design has been widely investigated to offer high interactivity and enhanced efficiency under limited device resources. Earlier researchers recognized that understanding human behaviors is an important task to accomplish such goals. For example, diverse techniques exploited human activities and contexts as key control knobs of various system managements including mobile systems [9] and smart homes [1].

Human activity recognition such as motion detection is a key part of these techniques. Machine learning (ML) techniques are often used to automatically identify the activities from various information, where devices in the loop need to collect the data using sensors, e.g., accelerometers and GPS. Since training ML models are computationally intensive tasks in general, the expensive tasks are often offloaded to powerful systems, e.g., clouds. However, the emergence of the Internet of Things (IoT) raises several issues in this approach; the amount of data created by billions of distributed devices adds significant computation burden to the centralized cloud. In addition, sending the sensitive user information may pose privacy and security concerns. An alternative solution is to run these tasks in a more localized way, e.g., on the IoT gateways at the edge [3, 19]. The local IoT devices typically have less computing resources than the cloud servers and run on low-power processors, such as ARM or Intel Atom. Thus, we need a new ML technique that can be efficiently processed even on the embedded devices.

Towards this goal, we have developed a new methodology which can efficiently recognize human activities based on hyperdimensional (HD) computing. HD computing is recently developed as an alternative computing method inspired by the human brain [8]. It represents the brain's memory model using data encoded at large dimensionality. Earlier works show that HD computing can offer high efficiency in many classification tasks, e.g., voice recognition [5] and language identification [7]. HD computing is in particular suitable for sensor-based classification tasks like human activity recognition in IoT devices since it is robust against most hardware failure mechanisms and thrives on noisy and incomplete data that the sensors often provide [6].

In this paper, we describe how the HD computing method can be applied to solve diverse human activity recognition problems. We model the classes of the human activities with high dimensional vectors, called *hypervectors*. Our approach encodes the collected sensor samples with hypervectors, and combines the samples for each class into a single hypervector using robust algebra in HD space. Since this step only uses simple operations, e.g., element-wise addition and multiplication, we can train the model in a lightweight way. Once the modeling is completed, we identify the human activity class for a newly observed data encoded with a hypervector. To this end, we search the most similar hypervector in the model to the sample.

We have designed different variants of the HD computing-based classification method for higher efficiency and classification accuracy. In this paper, we present two key approaches, *hypervector retraining* and *hypervector binarization*. The hypervector retraining refines the models to achieve higher classification accuracy. Unlike previous work, in the retraining step, we exploit non-binarized hypervectors to achieve high accuracy. The hypervector binarization then converts the trained hypervectors back to hypervectors of bitstreams, making the HD computation more suitable for less-powerful IoT devices.

In our evaluation, we compare our approach with the state-of-the-art ML solutions. Our experimental results show that

This work was supported in part by CRISP, one of six centers in JUMP, an SRC program sponsored by DARPA and NSF grants #1730158 and #1527034, and Jacobs School of Engineering UCSD Powell Fellowship.

IOT '18, October 1518, 2018, Santa Barbara, CA, USA 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-6564-2/18/10$15.00
`https://doi.org/10.1145/3277593.3277617`

the proposed method can provide high accuracy and computing efficiency for popular human activity recognition problems. For example, as compared to the neural networks-based modeling, the HD-computing method achieves up to 486x speedup when running on x86 processor, while providing comparable classification accuracy. In addition, our design improves the performance of HD model-based inference tasks by up to 7x on a low-power ARM processor.

The rest of the paper is organized as follows: We first discuss related work for HD computing, and elaborate on the background of HD computing. Next, we describe the HD-based solution for human activity recognition with our strategies to optimize the efficiency and accuracy. The next section presents experimental results, and we conclude the paper with a discussion for future work.

RELATED WORK

Human Activity Recognition

Prior researchers have been investigated to understand and identify human activities and contexts. For example, the work in [16] showed a monitoring framework for human activity recognition which collects data from inertial measurement units (IMU). Some works have shown that daily activities can be captured by the sensors equipped in smartphone systems [2, 17]. Another line of research has focused on how to exploit the human activity and context information for diverse problems. For example, prior research has shown that understanding users behavior and exploiting the behavioral characteristics can be used to improve system efficiency. In this context, earlier work proposed diverse system optimization techniques by identifying user behaviors and interactions for mobile systems [9] and smart homes [1]. Prior work often utilized ML techniques to identify the activities, while relying on computing capability of clouds through offloading, e.g., [18]. However, due to the massive data stream created in the IoT systems, more light-weight alternatives are considered as a key requirement in the system design.

HD Computing

The hyperdimensional (HD) computing was first introduced in the field of neuroscience [8]. Prior researchers recognized that HD computing is effective for pattern-based cognitive tasks, and showed diverse applications, such as language recognition [7], text classification [11], the prediction from multimodal sensor fusion [14, 15], and speech recognition [5]. The work in [13] showed that bio signal sensory data can be represented with hyperdimensional data. Some work have also presented that HD tasks can be efficiently performed with diverse computing devices. For example, the hardware accelerator design has been proposed to efficiently compute binarized hypervectors. Some works also presented new memory architectures that perform HD operations inside memory arrays [10, 6]. Digital circuits for HD computing have been also designed, e.g., computation of Hamming distance distance search [6]. In this paper, we focus on how the human activity recognition problem can be effectively mapped using HD computing. In addition, we show how the HD computing can be further optimized for IoT devices.

BACKGROUND: HD COMPUTING

In this section, we discuss the background for HD computing, focusing on the components used in our activity recognition design.

Data type: Unlike conventional computing methods, the basic data type of the HD computing is the hypervector which often has many elements, e.g., more than one thousand. We denote the dimensionality of the hypervector using D. For example, collected data are converted to hypervectors for future HD procedures, e.g., classification tasks. In the earlier HD work, e.g., [5], each element of hypervector is assumed to be a bit. In contrast, since the hypervector containing numbers may include diverse information, some recent HD applications choose this data type to implement [4]. We call these two different types as *binary* hypervector and *non-binary* hypervector. An element of a binary hypervector can be either 0 or 1. For the non-binary hypervector, the elements can have any real number.

Property of hypervectors: An important characteristic used in HD computing is the orthogonality of hypervectors. Let us assume that there are two hypervectors, A and B. The non-binary hypervectors are defined to be orthogonal if the cosine similarity of A and B is zero. For binary hypervectors, we can define the orthogonality by mapping the hypervector element of 0 to -1.

Since a hypervector has a large number of elements, we can easily find many pairs of two orthogonal hypervectors by randomly selecting their elements. For example, let us assume that we randomly choose elements of two non-binary hypervectors, A and B, among -1 and 1. In the cosine similarity computation, the element-wise multiplication make each element to either -1 or 1 with 50% chance, and the summation of all elements are very close to zero, i.e., near orthogonal. In contrast, if two hypervectors are computed somehow to be similar, the cosine similarity has a high value.

HD arithmetic operations: HD arithmetic operations enable to associate multiple hypervectors. In this paper, we utilize three major operations.

- Binding: Two hypervectors A and B are combined into a hypervector. We denote this operation with $A \times B$. For the binary hypervectors, the element-wise XORing accomplishes this procedure; the element-wise multiplication is used for non-binary hypervectors. The binding operation preserves orthogonality of hypervectors. For example, when we have three hypervectors randomly created, say X, Y, and Z, the hypervector X is still near-orthogonal to the binding of the rests, $Y \times Z$.

- Bundling: This operation is denoted with the $+$ symbol. The component-wise addition implements the bundling for non-binary hypervectors. Since the bundling for two binary hypervector yields a non-binary hypervector, a majority function is applied afterward. For example, when n binary hypervectors are bundled, we first apply the elements-wise addition, and make each element whose value is greater than $n/2$ to 0; 1 for the other case. We denote this operation by $[A_0 + A_1 + \cdots + A_n]$. The bundling operation preserves the similarity with the combined hypervectors. For example, for two hypervectors A and B,

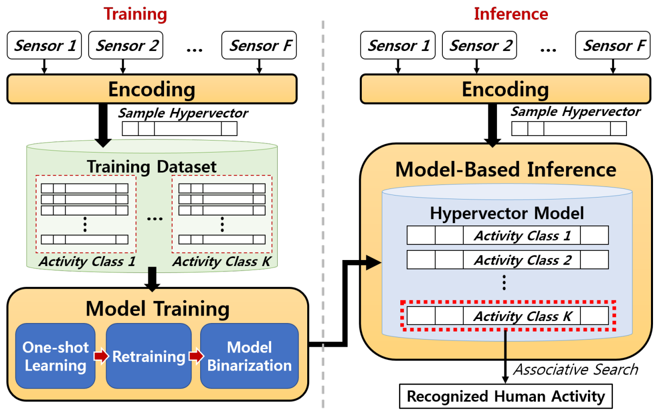


Figure 1. Overview of Proposed Design

the cosine similarity between A and $A+B$ is $cos(\pi/4)$, i.e., greater than zero.

- Detaching: This is a counter operation of the bundling for non-binary hypervectors. The component-wise subtraction implements this operation, and we denote it using the $-$ symbol. This makes the cosine similarity between two operand hypervectors either smaller or negative.

Associative search: The binary and non-binary hypervectors respectively use Hamming distance and cosine distance as their distance metrics. For simplicity, we denote the distance metric, which is appropriate for each case, by $\delta(A,B)$. When we have multiple hypervectors, the associative search is used to find the most similar hypervectors using the distance metric. For example, when we have m hypervectors, $H_1, \cdots, H_m$, the associative search for another hypervector A looks for a hypervector, H_i, whose $\delta(H_i, A)$ is the highest.

HD-BASED HUMAN ACTIVITY RECOGNITION

Design Overview

Figure 1 describes our design that recognizes human activities based on HD computing. We collect multiple raw data from the external sensors in IoT devices, e.g., IMUs of wireless embedded devices and accelerometers in smartphones. Instead of using real numbers, we convert each collected sample, which includes multiple measurements, to a hypervector. We call this step by *encoding*. With the encoded hypervectors and its original activity (label), e.g., walking, running, and standing, we train the hypervector model. To classify K classes, the trained model includes K hypervectors for each class. The training procedure consists of three parts, one-shot learning, retraining, and model binarization. In the one-shot learning, our design reads and process the hypervectors for each sample one by one. Then, the retraining refines the hypervector models considering the samples again with multiple iterations. In the next step, we update the model to the binarized hypervectors for performance improvements. With the trained model, we perform the inference of the class. The goal of the inference procedure is to classify a collected sample with an unknown label into an activity class. Our design accomplishes the inference by performing the associative search with the model hypervectors.

Sensor Data Encoding

To enable HD computing, we encode the collected raw data to hypervectors. Let us assume that a sample collected at a time includes F values, i.e., $S = \langle v_1, \cdots, v_F \rangle$, where each v_i is different raw values that each sensor measures. To find the patterns of sample hypervectors for each human activity, the encoding procedure considers the impact of i) the value for each sensor measurement and ii) differences of all the sensors in the system.

The first step of the encoding is to convert a measurement value, v_i, into a hypervector. As discussed in the background section, the similarity between two hypervectors, A and B, is determined with a metric, i.e., $\delta(A,B)$. Thus, we encode each value so that the corresponding hypervector keeps the relative difference across the measurement values of different samples under the distance metric. To this end, we utilize the measurement range of each sensor. For example, if a sensor produces a value in a range of $[V_{min}, V_{max}]$, the minimum and maximum values correspond to two hypervectors, L_{min} and L_{max}, where L_{min} and L_{max} are orthogonal to each other.

We represent any measurement value using the two hypervectors. L_{min} with D dimension is first created by randomly choosing its elements. Using the L_{min}, we create another hypervector, say L_1, by flipping $D/2Q$ elements, where Q is a configurable value. We repeat this procedure by Q times to decide $L_1, L_2, \cdots, L_Q$, e.g., flipping elements of L_1 creates L_2. Note that L_Q is orthogonal to L_{min}, thus $L_Q = L_{max}$. We call these created hypervectors as *level hypervectors*. A level hypervector corresponds with each measurement value by considering the relative difference of the measurement values. To this end, where the measurement range is quantized to Q levels, and each quantized subrange is mapped to a level hypervector.

In the second step of the encoding, we combine different sensor values of a sample to represent it with a single hypervector. To distinguish different sensors in the hypervector representation, we utilize another set of hypervectors, $B_1, \cdots, B_F$, called *base hypervectors*, whose elements are randomly chosen for the orthogonality. Let assume that each v_i value corresponds to a level hypervector, L_i. The encoded hypervector for the sample is computed by:

$$H = L_1 \times B_1 + \cdots + L_F \times B_F.$$

Since the B_i hypervectors are orthogonal, even though we use the same set of the level hypervectors for different sensors, our training step still distinguishes the impact of different sensors within the encoded hypervector. All of the random hypervectors, i.e., L_{min} and B_i, are required to be created only once and exploited for the entire recognition procedure. Note that the elements of the encoded hypervectors, say *sample hypervectors*, are 0 or 1 if using the binary hypervectors; -1 or 1 for the non-binary hypervector case.

Model Training

In this procedure, our design trains the model by combining the sample hypervectors. The goal is to learn the patterns of sensor values which exist within a class. Let assume that the training dataset includes N samples, and each sample is

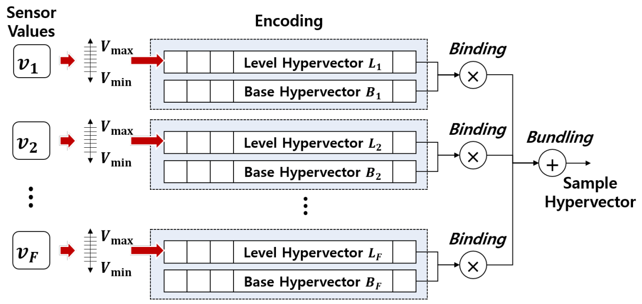


Figure 2. Encoding of Sensor Measurements

encoded with N hypervectors, $H_1, \cdots, H_N$. Each sample hypervector corresponds to an activity class, say c_i.

One-shot training: The first step of the training is to bundle the hypervectors for each class. We call this computation as *one-shot* training. For example, let us assume that there are l hypervectors, $H_1, H_2, \cdots, H_l$, where all of them are included in the same class. The bundling operation makes another hypervector, $M = H_1 + \cdots + H_l$. For example, let us assume that we have another hypervector H_{test}, which is very similar to H_1, by the distance metric. In this case, $\delta(M, H_{test})$ is likely to be a positive value. Furthermore, if H_{test} is similar to the majority of the hypervectors combined into M, $\delta(M, H_{test})$ yields a much higher value. Based on this observation, we create the one-shot model, say $M_1, \cdots, M_K$, by bundling all sample hypervectors included in each activity of K classes.

Retraining: An issue of the one-shot model is that, although the bundled hypervectors captures the major similarity *within* each class, it does not understand hypervector differences *across* classes. In addition, bundling a large number of hypervectors may degrade classification quality when a large variety of patterns exists in each class. Thus, we refine the model to i) better identify the discrepancy between different classes and ii) recognize the common pattern existing in each class.

Algorithm 1 illustrates our retraining procedure to reduce the misclassification rate of the activity recognition. From the one-shot model, our design verifies the classification accuracy for each sample using the associative search. If a sample is wrongly classified, we modify two model hypervectors, i.e., the hypervector of the target class and the other hypervector of the misclassified class. We first bundle the sample hypervector once more to the correct class so that the model hypervector converges faster to the misclassified sample. The second task is detaching the hypervector from the wrong class to enlarge the difference between the two model hypervectors. We repeat this updating process multiple times for the training dataset, and the accuracy converges with sufficient iterations.

Model Binarization: Since our model retraining algorithm exploits the element-wise addition and subtraction in the bundling and detaching operations, it consequently creates non-binary hypervectors as the model. Even though it makes the model more accurate, the model size and computation costs of the inference also increase. Since many devices

Algorithm 1: Pseudo code of retraining precedure

```
$t \leftarrow 0$
while $t <$ *# of Iterations* do
    $t \leftarrow t + 1$
    for *each sample hypervector, $H_i$* do
        $\rho \leftarrow$ associative search for $H_i$ in the model
        if $\rho \neq c_i$ then
            $M_{c_i} \leftarrow M_{c_i} + H_i$
            $M_\rho \leftarrow M_\rho - H_i$
        end
    end
end
```

in IoT environments which run the activity recognition is less-powerful, we optimize the model by converting the model to the binary hypervectors. We update the model depending on the sign of each hypervector element, i.e., choosing 1 if the element value is positive; 0 for the negative value.

Model-Based Inference

Once the model is trained, it is ready to process the inference step for samples whose labels are unknown. We first encode the values using the level and base hypervectors used in training step. Then, our design finds which model hypervectors is the most similar to the given sample hypervector using the associative search. Note that, in the associative search, we use different distance metrics based on the data type of the model. In general, the non-binarized model provides better accuracy. In contrast, the binarized model processes the inference in a more efficient way, since the Hamming distance can be computed with bitwise XOR operations for the smaller model, unlike the element-wise integer additions for the cosine distance computation.

EVALUATION

Experimental Setup

To evaluate how the proposed design works on the heterogeneous IoT environment, we utilize two different devices running on 2.8 GHz Intel Core i7 (x86) and 1.4 GHz ARM Cortex-A53 (ARM) processors. For both cases, we execute the same code implemented with Python 2.7 and Numpy which uses C++ backend. We compare our approach with the state-of-the-art deep neural network models (DNN) implemented using Google TensorFlow. Since our design can create binarized hypervector models, for fair comparison, we also evaluate the binarized neural network (BNN) models. The neural network models have three hidden layers of 512 neurons, and DNN and BNN models are trained with ADAM optimizer for 10 and 100 epochs, respectively, so that the accuracy converges. For the efficiency comparison, we measure the execution time of the training and testing procedures.

We evaluate our approach using three practical datasets as follows.

UCIHAR: This dataset includes the sensor measurements for accelerometers and gyroscopes of a smartphone, which are measured on the waist of users. The goal is to classify twelve activity classes, e.g., walking, walking up/downstairs, sitting and standing.

Name	Data Size	F	K	N_{train}	N_{test}
UCIHAR [2]	10MB	561	12	6213	1554
PAMAP2 [16]	240MB	75	5	611142	101582
EXTRA [17]	140MB	225	4	146869	16343

Table 1. Evaluated Dataset (F: the number of features, K: the number of activity classes, N_{train}: the number of samples in the training dataset, N_{test}: the number of samples in the testing dataset)

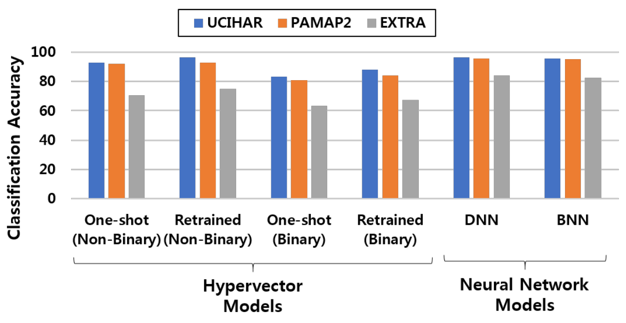


Figure 3. Accuracy Comparison for Different Modeling Methods

PAMAP2: The dataset contains data measured from three IMUs located at the wrist, chest, and ankle of users with a heart rate monitor. The goal is to classify five basic activities, e.g., walking and sitting. We exploit the feature extraction method suggested by the author.

EXTRA: The dataset has measurements of heterogeneous sensors from smartphones and smartwatches. We choose to classify the activity labels for phone locations, e.g., whether it is located on the table, in the pocket, bag, and hand. Note that the activities are related to diverse device control problems, e.g., thermal management of mobile devices [12].

Table 1 summarizes the dataset sizes. In our evaluation, we set the quantization level to 8, the retraining iterations to 20, and the dimension of hypervectors to 1000, since there is no accuracy gain with larger values.

Classification Accuracy

Figure 3 shows the comparison results of the accuracy for different modeling methods. The results show that the proposed retraining method improves the classification accuracy. For example, when using the non-binary hypervector models, the accuracy improvement is 3% on average. We observe higher accuracy improvements for the binarized hypervector models by 4% on average. Throughout the retraining procedure, we train the HD model which have comparable accuracy to the DNN and BNN models. For example, for UCIHAR dataset, the accuracy difference between the non-binary model and DNN is only 0.2%. The accuracy difference between binary and non-binary models is 8% on average. In the next section, we evaluate how much performance can be improved by the model binarization.

Efficiency Comparison

Training Efficiency

We evaluate the efficiency of different modeling methods. Figure 4(a) shows the efficiency comparison of our design with the state-of-the-art DNN and BNN model. The results are reported for the non-binarized models, since the overhead of the model binarization is negligible.[1] In this comparison, the HD modeling and the neural network training were both executed on x86 processor. The results show that the proposed method presents higher performance efficiency as compared to the neural network training. For example, for UCIHAR dataset, training the HD model with the retraining procedure is 4x and 56x faster than the DNN and BNN models, respectively. Note that the accuracy difference between the two model is only 0.2% as presented in the previous section.

In addition, when a small amount of accuracy loss is acceptable, our design can also train the model without retraining. In that case, we observe the speedup up to 486x compared to the BNN approach.

Figure 4(b) compares the execution time of the training procedure on the two different processors. The results suggest that the proposed design can efficiently train the hypervector model even on the low-power processor. For example, for PAMAP2 dataset, the training time including the retraining only takes 26 seconds on the ARM processor. To train the one-shot model, it only takes 4 seconds. Thus, we conclude that the proposed design may efficiently process the activity recognition tasks in the IoT systems, since many IoT devices in the loop is expected to run on low-power processors with resource budgets.

Inference Efficiency

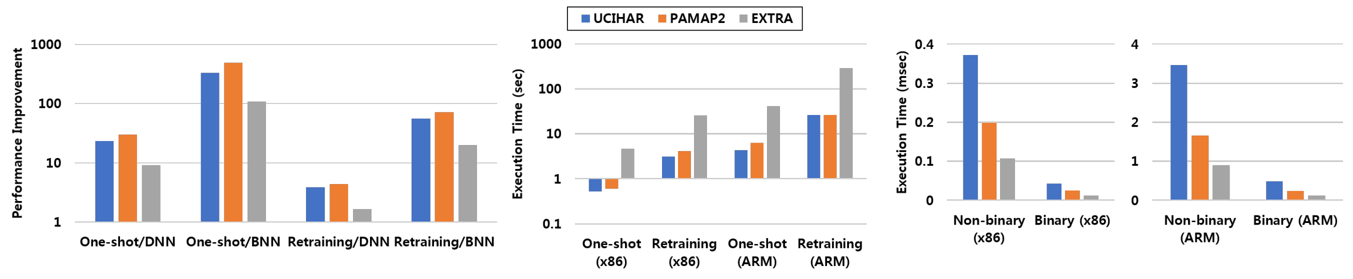
With the trained model, our design performs the inference tasks for each collected data. Figure 4(c) shows how much the execution time takes to process the inference procedure for each sample. In this evaluation, we compare the non-binary model to the binary model. The result shows that the model binarization significantly improves the inference procedure. The speedup is 8.4x and 7.1x for the x86 and ARM case, respectively.

For the ARM processor case, the inference based on the non-binarized model takes 2 ms on average, while the binarized model only takes 0.28 ms. In IoT systems, the sensors are often equipped with the same device running on these low-power processors. Thus, when a small amount of accuracy loss is acceptable, the binarized model is more preferable, e.g., serving real-time needs for the activity recognition.

CONCLUSION

In this paper, we present a new system design which performs human activity recognition tasks based on emerging HD tasks. We also show optimization techniques that improve the accuracy of the HD-based classification and performance efficiency in the inference procedure. In our evaluation, the proposed design achieves the speedup of the model training by up to 486x, compared to the neural network model training. The proposed work can be extended to several directions. The HD method can be also applied for other classification tasks existing in IoT systems. We plan to investigate further optimization strategies to improve the HD classification quality. In addition, we also work on hardware design for the proposed method in order to more efficiently exploit the HD computing in control systems of low-end devices.

[1]The model binarization requires to update the trained hypervectors only once after all the retraining procedure.



(a) Comparison with DNN and BNN (Training) **(b) Execution time in HD Model Training** **(c) Execution time in HD Model-Based Inference**

Figure 4. Efficiency Comparison for Training and Inference

REFERENCES

1. Aksanli, B., Akyurek, A. S., and Rosing, T. S. User behavior modeling for estimating residential energy consumption. In *Smart City 360*. Springer, 2016, 348–361.
2. Anguita, D., Ghio, A., Oneto, L., Parra, X., and Reyes-Ortiz, J. L. A public domain dataset for human activity recognition using smartphones. In *ESANN* (2013).
3. Cui, W., Kim, Y., and Rosing, T. S. Cross-platform machine learning characterization for task allocation in iot ecosystems. In *Computing and Communication Workshop and Conference (CCWC), 2017 IEEE 7th Annual*, IEEE (2017), 1–7.
4. Imani, M., Huang, C., Kong, D., and Rosing, T. Hierarchical hyperdimensional computing for energy efficient classification. In *Proceedings of the 55th Annual Design Automation Conference*, ACM (2018), 108.
5. Imani, M., Kong, D., Rahimi, A., and Rosing, T. Voicehd: Hyperdimensional computing for efficient speech recognition. In *International Conference on Rebooting Computing (ICRC)*, IEEE (2017), 1–6.
6. Imani, M., Rahimi, A., Kong, D., Rosing, T., and Rabaey, J. M. Exploring hyperdimensional associative memory. In *High Performance Computer Architecture (HPCA), 2017 IEEE International Symposium on*, IEEE (2017), 445–456.
7. Joshi, A., Halseth, J., and Kanerva, P. Language geometry using random indexing. *Quantum Interaction 2016 Conference Proceedings* (In press).
8. Kanerva, P. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive Computation 1*, 2 (2009), 139–159.
9. Kim, Y., Parterna, F., Tilak, S., and Rosing, T. S. Smartphone analysis and optimization based on user activity recognition. In *Computer-Aided Design (ICCAD), 2015 IEEE/ACM International Conference on*, IEEE (2015), 605–612.
10. Li, H., Wu, T. F., Rahimi, A., Li, K.-S., Rusch, M., Lin, C.-H., Hsu, J.-L., Sabry, M. M., Eryilmaz, S. B., Sohn, J., et al. Hyperdimensional computing with 3d vrram in-memory kernels: Device-architecture co-design for energy-efficient, error-resilient language recognition. In *Electron Devices Meeting (IEDM), 2016 IEEE International*, IEEE (2016), 16–1.
11. Najafabadi, F. R., Rahimi, A., Kanerva, P., and Rabaey, J. M. Hyperdimensional computing for text classification. *Design, Automation Test in Europe Conference Exhibition (DATE), University Booth* (2016).
12. Paterna, F., and Rosing, T. Š. Modeling and mitigation of extra-soc thermal coupling effects and heat transfer variations in mobile devices. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, IEEE Press (2015), 831–838.
13. Rahimi, A., Benatti, S., Kanerva, P., Benini, L., and Rabaey, J. M. Hyperdimensional biosignal processing: A case study for emg-based hand gesture recognition. In *Rebooting Computing (ICRC), IEEE International Conference on*, IEEE (2016), 1–8.
14. Räsänen, O., and Kakouros, S. Modeling dependencies in multiple parallel data streams with hyperdimensional computing. *IEEE Signal Processing Letters 21*, 7 (2014), 899–903.
15. Rasanen, O., and Saarinen, J. Sequence prediction with sparse distributed hyperdimensional coding applied to the analysis of mobile phone use patterns. *IEEE Transactions on Neural Networks and Learning Systems PP*, 99 (2015), 1–12.
16. Reiss, A., and Stricker, D. Introducing a new benchmarked dataset for activity monitoring. In *Wearable Computers (ISWC), 2012 16th International Symposium on*, IEEE (2012), 108–109.
17. Vaizman, Y., Ellis, K., and Lanckriet, G. Recognizing detailed human context in the wild from smartphones and smartwatches. *IEEE Pervasive Computing 16*, 4 (2017), 62–74.
18. Yan, T., Chu, D., Ganesan, D., Kansal, A., and Liu, J. Fast app launching for mobile devices using predictive user context. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, ACM (2012), 113–126.
19. Yi, S., Li, C., and Li, Q. A survey of fog computing: concepts, applications and issues. In *Proceedings of the 2015 workshop on mobile big data*, ACM (2015), 37–42.